

Software Engineering

Group 63



Design of a sustainable transport rewards application

Rory Simpson - 2454453 - rxs1099
Daniel Lawson - 2428894 - dxl295
Allen Jen Joseph - 2549623 - ajj223
Caleb George - 2321112 - ceg212
Frederick Foster - 2429087 - fjf287
Conor Galvin - 2431008 - cxg112
Daniel Jones - 2456299 - dxj299
Lilliana Etheridge - 2433639 - lxe239

A1: Introduction

Introduction

This project involves designing a mobile app to incentivize sustainable transportation, encouraging users to walk, cycle, or use public transport. Two subsystems track user behaviour: one awards points based on walking or cycling distance, and the other awards points for entering public transport ticket codes or scanning tickets. Users spend points on sticker packs to fill a virtual album, promoting a gamified approach to sustainable behaviour. Unlike the similar app Strava, our focus is on environmental sustainability, encompassing various modes of eco-friendly transport beyond exercise.

Assumptions

The following are some assumptions we have made about operations and systems relating to our system:

- The system can access data from the mobile device's GPS receiver, accelerometer, and camera.
- By partnering with the transport companies, the system will be able to check if a ticket is legitimate when scanning it, either via the camera or entering a ticket's code.
- By partnering with the transport companies, the system will be able to determine the difference between a scanned bus/tram/train ticket and a bus/tram/train pass
- By partnering with the transport companies, their databases will be able to keep track of if a ticket has already been used by another user.
- By partnering with the transport companies, the system will be able to use the mobile device's GPS tracker in conjunction with data from the transport company to check distance travelled along a public transit route
- The system will be able to communicate with the same system on a separate mobile device through the use of Wi-Fi.

Specification

Language selection

The app supports multiple languages, including English, Spanish, Chinese (Mandarin), Urdu, Punjabi, Bengali, and Arabic. Users can choose their preferred language upon opening the app, either by selecting directly or using a search bar. The interface is then updated accordingly. This diverse language support aims to enhance accessibility and encourage a broader user base to adopt sustainable practices. Selected languages are based on global significance and specific relevance to the city of Birmingham.

Viewing user profile

The user can open their profile at any time to view their progress, such as their current points or the percentage of unique stickers from the album that they have collected. This menu also features a share button, which allows the user to share their progress to major social media platforms such as Instagram and Twitter without the need to take a screenshot and leave the app. This menu also houses the option to change the display language. This menu is needed so that the above options don't clutter the main page, making the app easier to use.

Tracking walking/cycling distance

The app can read input from the phone's accelerometer and GPS receiver to measure the user's speed. It uses this information to determine if the user is walking or cycling and, if so, tracks the distance they are travelling. This can be done in the background even while the app is closed or when the phone is locked, so the user doesn't need to focus on the app at all times while walking. Then, when the user opens the app the next time they can claim the points they earned by walking. This is the same method that other apps that track distance such as Pokémon Go use, and it is suited to this system as well as it allows the user to gain the correct amount of points from walking without needing the app open at all times.

Scanning tickets

The app allows users to scan public transport tickets via their phone camera or manually input the ticket ID. It verifies ticket validity and rewards points based on expected distance, with different rates for various modes (e.g., trains earn more points than buses due to lower emissions). This is necessary to accurately give points for public transport usage, as just using GPS tracking would also allow points to be gained for car travel.

Redeeming sticker packs

Users can redeem points for stickers in the "sticker store" with three options: "purchase a sticker", "purchase a sticker pack" and "purchase a fresh sticker pack". Sticker packs, costing less than buying individual stickers, may include one unobtained sticker. A fresh pack, costing more, guarantees at least one new sticker. This randomness, akin to "Gacha", promotes prolonged user engagement. Notably, the system excludes real-money transactions, motivating users to sustain eco-friendly transport for sticker acquisition.

Trading stickers

Users can trade stickers through online servers, allowing them to exchange duplicates for new, unique stickers. One user initiates a trade, offering stickers, and the recipient can accept or decline. Accepted trades involve offering stickers in return, and the initiating user confirms or cancels the trade. If a user offers a sticker the other doesn't own, it appears in silhouette. Trading fosters a social aspect, promoting efficiency in completing the sticker album. This encourages users to engage friends in the app, creating a positive feedback loop for sustained use of eco-friendly transport.

Viewing stickers

The user can view their sticker collection in its entirety from their profile. They are organised into pages and can be filtered by criteria such as 'date earned' or 'rarity' (i.e. percentage of users who own this sticker). Pressing on a specific sticker icon will show an expanded version of the image for easier viewing. From this menu, the user can also share what stickers they have and their percentage completion to social media sites such as Instagram. This is needed to increase the social aspect of the app.

A2: Main requirements

Functional:

1. The system must provide users with different means of interaction with the system
 - 1.1. The system must be able to process commands from a touchscreen input.
 - 1.2. The system must be able to process commands from a keyboard input
 - 1.3. The system must be able to process commands from a camera input
 - 1.4. The system must be able to process input from the phone's built-in GPS receiver
 - 1.5. The system must be able to process input from the phone's built-in accelerometer
 - 1.6. The system could provide audio alerts (such as notification sounds) through a phone's speaker to get the customers' attention
 - 1.7. The system could provide notifications (either push notifications or rich notifications) to prompt the user to use the app more
2. The system should allow users to choose a language
 - 2.1. The system should provide the ability for users to choose their language on a multi-select screen
 - 2.2. The system should provide support for the English language
 - 2.3. The system should provide support for the Spanish language
 - 2.4. The system should provide support for the Chinese language
 - 2.5. The system should provide support for the Urdu language
 - 2.6. The system should provide support for the Punjabi language
 - 2.7. The system should provide support for the Bengali language
 - 2.8. The system should provide support for the Arabic language
3. The system must be able to record environmental activities
 - 3.1. The system must be able to determine, based on accelerometer input, if the user's device is moving at walking or cycling speed
 - 3.2. The system must detect how far the user has walked
 - 3.3. The system must detect how far the user has cycled
 - 3.4. The system must allow a user to scan bus tickets / passes onto the system via the device's camera

- 3.5. The system must allow a user to scan train tickets onto the system via the device's camera
 - 3.6. The system must allow a user to scan tram tickets / passes onto the system via the device's camera
 - 3.7. The system must allow a user to input a code to load a bus ticket / pass onto the system
 - 3.8. The system must allow a user to input a code to load a train ticket onto the system
 - 3.9. The system must allow a user to input a code to load a tram ticket / pass onto the system
 - 3.10. The system should be able to determine, using the mobile device's GPS tracker, how far the user has travelled on public transport
4. The system must allow the user to gain points
 - 4.1. The system must grant the user 1 points after recording 100 metres walked
 - 4.2. The system must grant the user 1 points after recording 120 metres cycled
 - 4.3. The system must grant the user 10 points after travelling 1 kilometre on a bus
 - 4.4. The system must grant the user 15 points after travelling 1 kilometre on a train
 - 4.5. The system must grant the user 12 points after travelling 1 kilometre on a tram
 - 4.6. The system must save the points earned by the user
 - 4.7. The system could grant the user 5 points for opening the app via a notification prompt to do so
5. The system must allow the user to collect virtual images, hereby referred to as stickers
 - 5.1. The system must allow users to exchange 100 points for a random sticker
 - 5.2. The system could allow users to exchange 400 points for a sticker pack, which contains 5 random stickers
 - 5.3. The system could allow users to exchange 700 points for a fresh sticker pack, which contains 5 stickers including at least one un-obtained sticker
 - 5.4. The system must save stickers acquired by the user
 - 5.5. The system must display a user's stickers in a virtual album within the application
6. The system should allow the user to interact with other users of the system
 - 6.1. The system should allow users to trade stickers with each other
 - 6.2. The system should allow the user to search for another user's username, and select them as the person to trade with.
 - 6.3. The system should allow the user to select which sticker(s) they want to send to another user
 - 6.4. The system should allow the user to send the sticker(s) they selected to another user
 - 6.5. The system should be able to remove the sticker(s) which has been sent to another user from the previous user's system
 - 6.6. The system should allow a user to receive stickers from another user
 - 6.7. The system should be able to add a sticker received from another user to the user's sticker album
7. The system could allow the user to share data with other applications

- 7.1. The system could allow the user to share their % completion of the sticker album to Instagram via a button within the app
- 7.2. The system could allow the user to share their % completion of the sticker album to Twitter via a button within the app
- 7.3. The system could allow the user to share their % completion of the sticker album to Facebook via a button within the app
- 7.4. The system could allow the user to share their % completion of the sticker album to WhatsApp via a button within the app

Functional Requirements prioritisation

Requirement ID	Priority	Reasoning
1.1 through 1.5	M	These different input methods are required not only so that the user can interact with the system, but also so that the system can take in the data needed for recording the transport methods. [1.1 to 1.3] are ways the user will interact within the app, while [1.4 to 1.5] record travel data outside of it.
1.6, 1.7	W	These would be additional ways of getting the user to interact with the system. Not necessary for the system itself to work, but would improve the chances of the user changing their lifestyle to a more environmentally sustainable one.
2 (2.1 through 2.8)	S	Multiple languages are very important to increase the possible user base of the system, encouraging more widespread behavioural change. Despite this, the priority is only Should as the system is designed for use primarily within the West Midlands where in the 2021 census 84.4% of households have English as the primary language (not even counting those who speak it as a second language)
3.1 through 3.3	M	Detecting the user's walking or cycling distance is the primary aspect of the system, and is required for other aspects such as the stickers to work.
3.4 through 3.9	M	The user being able to input their public transport tickets is equally as important as the walking/cycling data, as to encourage behavioural change as many different methods of sustainable living should be provided. Both the camera and code inputs are must-haves to make the system as flexible as possible to all scenarios, and by providing multiple inputs it lets the user interact with the system in an unobtrusive way, which is important to make them more likely to live sustainably.

3.10	S	Being able to track distance travelled on public transport is not strictly required for the app to function, however having it will allow for more accurate distribution of points to the user (preventing someone from “cheating the system” to avoid having to live sustainably) Without this, someone travelling a short distance on a bus would gain the same number of points travelling a further distance.
4.1 through 4.6	M	These are required as the actual incentive to take more sustainable transport methods. Without these, the system would just be recording distance travelled, which many existing systems already do.
4.7	W	Like the notifications themselves [1.6 / 1.7], gaining points via opening the app could encourage the user to interact with it more, but is not essential to the core system.
5.1, 5.4, 5.5	M	Being able to trade points for stickers is the final part of the core “game loop” encouraging behavioural change. Being able to then view all saved stickers in the gallery will let users compare with other users, as well as encourage them to collect even more stickers (which leads to them living more sustainably)
5.2, 5.3	W	These provide the user with more choice when it comes to buying stickers, which will create a better game experience, however they are not essential to the core experience.
6 (6.1 through 6.7)	S	Being able to trade stickers with other users will build a sense of community, as users will want to trade with each other to complete their albums. Having a community will encourage users to live sustainably because they will want to interact with other users more. These requirements are only Should rather than Must because although the benefits from inter-user interaction are great, the core experience doesn't require it.
7 (7.1 through 7.4)	C	These are other ways that users can help build a community, however, they are not integral to the core experience. Even without this feature built into the system itself, users who wanted to share progress would just screenshot their gallery, so this is only at Could priority.

Non-functional:

1. Efficiency (performance) - The system must respond within a reasonable amount of time
 - 1.1. The system must use less than 20% of a device's processing power while running as a background process
 - 1.2. The system must scan tickets (bus, train, or tram) within 5 seconds
 - 1.3. The system must show the updated number of points within 5 seconds of them being granted
 - 1.4. The system must show the sticker album within 5 seconds of it being opened
 - 1.5. The system must show the updated sticker album within 5 seconds of a sticker being bought, or traded with another user
 - 1.6. The system must handle increasing numbers of accounts without a loss in performance
2. Efficiency (space) - The system must only use a reasonable amount of storage space on the user's device
 - 2.1. The system must occupy less than 10% of a device's memory while running as a background process
 - 2.2. The system's total size must be no more than 500MB
3. Usability - The system must be easy to use and navigate for all users
 - 3.1. A new user should learn all features of the app within 1 minute of use
 - 3.2. A new user should learn how to navigate to all areas of the app within 1 minute of use
 - 3.3. The system should have a simple graphical user interface with as few options per screen as possible
 - 3.4. The system's menus should have a consistent look and feel that is not confusing
 - 3.5. The system must provide a "help" option on every screen that explains the operations that can be performed from that screen
 - 3.6. The system must be accessible to those with disabilities and follow accessibility guidelines such as WCAG
4. Reliability - the system must work as much as possible without bugs
 - 4.1. The system must achieve an MTBF (mean time between failures) of 96 hours
 - 4.2. The system must allow a way for users to report any issues with the system
 - 4.3. If an issue is reported, the system must be fixed within 24 hours
 - 4.4. Regular data backups to the cloud could be implemented in case of data loss
 - 4.5. The system should retain essential features of the app if the user is not connected to the internet/mobile data
5. Interoperability
 - 5.1. The system must correctly integrate with the systems of the mobile device's operating system
 - 5.2. The system should ensure that all user interaction has cross-platform compatibility between different mobile operating systems, namely iOS and Android
6. Legislative (Privacy)

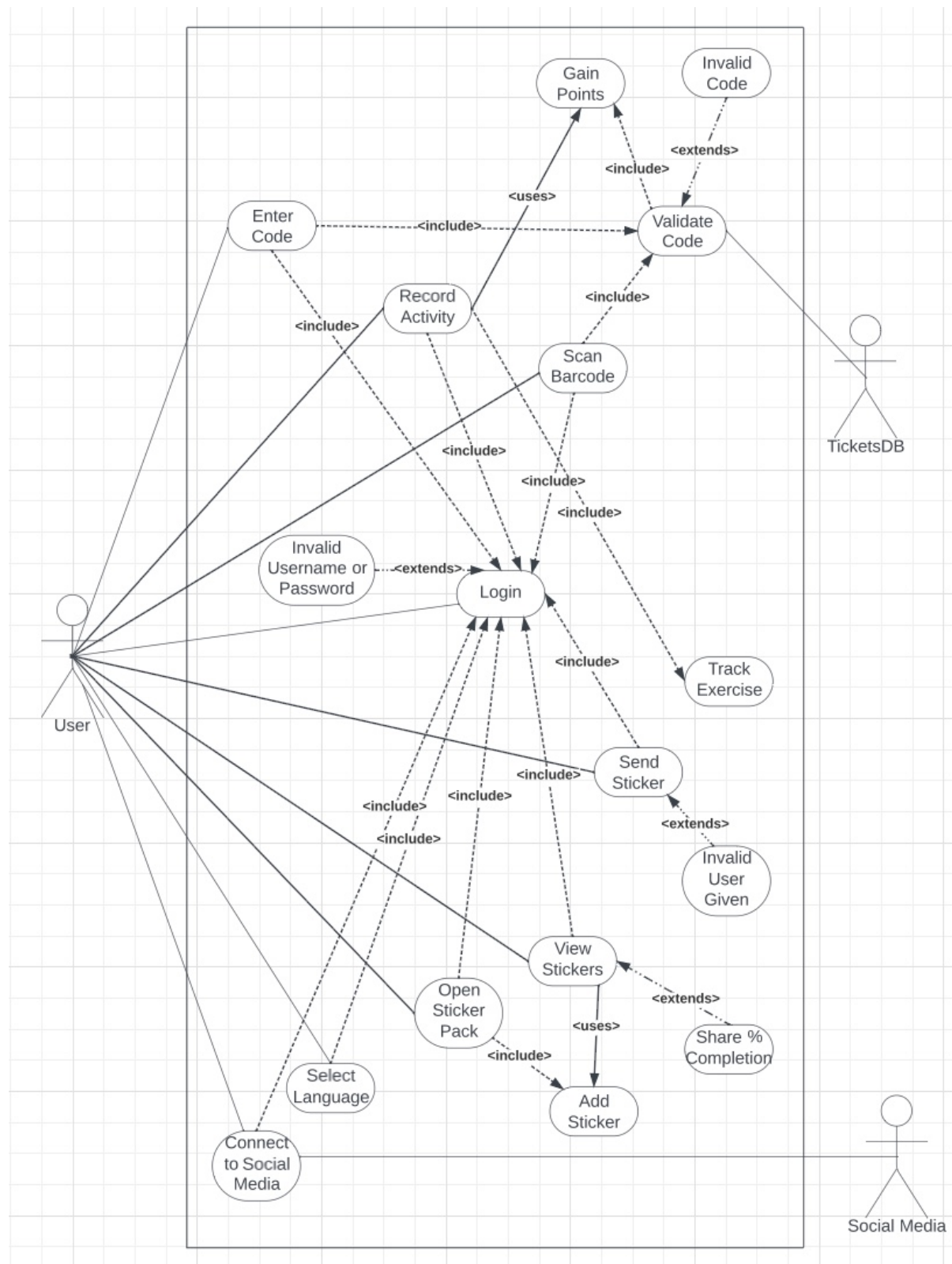
- 6.1. A privacy policy must be implemented to comply with GDPR, DPA and can be accessed at any time from the main menu
 - 6.2. Clear terms and conditions must be set out for app usage and can be accessed at any time from the main menu
- 7. Legislative (Safety)
 - 7.1. Users must be able to report any safety concerns and violations in the app
 - 7.2. The system must be able to handle reports appropriately
 - 7.3. The system must follow current health, safety and environmental regulations
 - 7.4. The system must have two-step verification to verify the identity of users
- 8. Ethical
 - 8.1. The system must only use the user's location to track distance and speed
 - 8.2. The system must correctly store and protect the user's account information to ensure it is not lost or tampered with
 - 8.3. The system must clearly communicate how the user's data is being used in the app
 - 8.4. The system must receive explicit permissions from the user before accessing their personal data and location
 - 8.5. The system must be accessible to all demographics and backgrounds
- 9. Delivery
 - 9.1. The system should be implemented and delivered by the end of the 2023 fiscal year.
 - 9.2. The system should be optimised upon listing on app stores with the use of keywords, images and descriptions
 - 9.3. The system should be tested thoroughly with selected users as a beta test before delivering the app to the wide-scale public
- 10. Implementation Constraints
 - 10.1. The system should be written in Java
 - 10.2. The system should be completed using an object-oriented approach
 - 10.3. The structure of the code should conform to good design practices, such as being written in a clear and simplistic manner
 - 10.4. The code should be sufficiently documented so that if any issues are reported it is easier to fix them
 - 10.5. The system should specify any hardware capabilities required for the app to function properly
- 11. Standards
 - 11.1. Information provided by the user when making an account must be sufficiently protected in accordance with the UK Data Protection Act 1988.

Non-Functional Requirements prioritisation

Requirement ID	Priority	Reasoning
1.1 through 1.6	M	The system must be as responsive as possible, if any actions take too long then users may be more likely to stop using the app, which would prevent our goal of promoting sustainable living.
2.1, 2.2	M	As with the performance efficiency, if the device takes up too much space in memory then the user is more likely to stop using it (such as if they need to delete an app to free space on their phone, they are more likely to delete this system if it takes up too much space)
3.1 through 3.4	S	These requirements all cover the initial usability of the system. The system must be easy to understand so that users do not get frustrated and choose instead to not use the system (as if they do so they would not change their behaviour). It is only at Should priority because of the following requirements [3.5]
3.5	M	Every screen needs to have a tutorial explaining the operations so that any user can understand the system, even those who are not familiar with similar mobile apps. This allows more people to use the system, promoting sustainable behaviour to more people.
3.6	M	To promote sustainable behaviour to as many people as possible, there must be as little barrier to such behaviour as possible. This is why having the system be as accessible as possible is important.
4.1 through 4.3	M	The system must be usable as much of the time as possible without flaws to successfully promote behavioural change. If users repeatedly encounter issues preventing them from engaging with the system, they are more likely to stop using it.
4.4	W	While storing user data in the cloud would be useful in some scenarios, it is not crucial to the core user experience.
4.5	S	By allowing the user to still gain points (from walking / cycling) and use those points to buy stickers while not connected to the internet will encourage them to open the app more often even while not at home, making them more likely to change their behaviour
5.1	M	If the system is unable to work properly on the mobile device then it will prevent it from running properly, causing the same issues as detailed in [4.1 through 4.3]

5.2	S	If users on different operating systems are unable to interact with each other, it will prevent the building of a community between users.
6 (6.1, 6.2)	M	User data must be handled carefully in line with the law and with the user's express permission. If this is not done then users may not trust the system, and so will stop using the app.
7 (7.1 through 7.4)	M	Like above [6.1, 6.2], it is important that users feel safe to use the app.
8.1 through 8.4	M	Like above [6.1, 6.2].
8.5	M	To promote widespread behavioural change, the potential user base of the system must be as large as possible. To this end, the system must be as accessible as possible no matter a person's demographic or background.
9.1	S	For a person's behaviour to change, it is important to have the change begin as soon as possible. Due to this, the system should be implemented as soon as possible (however the system's quality should not be sacrificed for the sake of this)
9.2	S	By optimising the app's pages on storefronts like the Play Store or App Store, more people will use the app and thus more people may begin living sustainably.
9.3	S	Performing a closed test will help ensure the system works as intended before it is released.
10 (10.1 through 10.5)	S	These requirements that detail the systems implementation are important but deviations from these will not cause major issues to the system itself. [10.4] is the most important, as it interlinks with [4.1 through 4.3]
11 (11.1)	M	Like above [6.1, 6.2]

B1: Use case diagram



B2/B3: Non-trivial and critical use cases

Use Case: Scan Barcode	
Actors	Primary Actor: User - initiator that interacts with the application, executing the Barcode/QR code scanning functionality, by employing the device's camera.
Preconditions	<ul style="list-style-type: none">• The user is logged into the app• The user has not scanned the same barcode before
Flow of Events	<ol style="list-style-type: none">1. The user selects the "Log Activity" option.2. The system activates the device's camera.3. The user scans the ticket Barcode/QR code using the device's camera.4. The system processes the code and checks if it is valid.5. If the Barcode/QR code is valid, the system records information on the scanned ticket.6. The system then accesses the record activity to assign points for the valid Barcode/QR code.7. The system adds the assigned points to the user's existing points tally.8. The user receives an in-app notification informing them of the number of points added.9. The user is redirected to the main menu, showing their updated points tally.
Alternative Flows	<p>Alternative Flow (Invalid Ticket)</p> <ol style="list-style-type: none">1. Invalid Barcode/QR Code:<ol style="list-style-type: none">5a: If the Barcode/QR code is invalid, the system alerts the user that scanning has been unsuccessful.6a: The system does not proceed with recording activity or assigning points.8a: The user is notified about the unsuccessful scanning attempt.
Postconditions	<ul style="list-style-type: none">• If successful, the system records ticket information and adds an appropriate number of points to the user's account• If unsuccessful, the system informs the user that scanning has been unsuccessful
Scenarios	<ol style="list-style-type: none">1. A user logs into the app and clicks the "Log Activity" option. The system then activates the device's camera which then scans the user's valid Barcode/QR code. The system then records the ticket information and assigns points to the user which are then added to the user's existing points tally.

	<ol style="list-style-type: none"> 2. A user logs into the app and clicks the “Log Activity” option. The system then activates the device’s camera which then scans the user’s invalid Barcode/QR code. The system then alerts the user with a message stating the scanning has been unsuccessful. 3. A user logs into the app and clicks the “Log Activity” option which then activates the device’s camera. The user decides to cancel the scan which makes the system abort the scan bringing the user back to the app’s main screen. 4. A user logs into the app and clicks the “Log Activity” option which then activates the device’s camera which then scans the user’s valid Barcode/QR code. The system encounters an error in processing and the user is informed about the situation mentioning that no points have been added.
--	---

Use Case: Track Exercise	
Actors	Primary Actor: User - Initiates the ‘Track Exercise’ process by using the app and performing physical activities which are tracked
Preconditions	<ul style="list-style-type: none"> • User has logged into the app for the first time. • User’s device is powered on. • The system has access to the device’s GPS, Accelerometer and Pedometer.
Flow of Events	<ol style="list-style-type: none"> 1. The “Track exercise” process starts and runs in the background. 2. The process records the following if/when they occur: <ol style="list-style-type: none"> a. The device’s pedometer detects some number of steps taken by the user. b. The device’s accelerometer detects the user moving at a speed recognized as walking/cycling. c. The device’s GPS detects that the user has moved some distance. 3. The User re-opens the app. 4. Based on information recorded in step 2 (if any), the app calculates an appropriate number of points to grant the user. 5. The calculated points are added to the user’s total. 6. The User receives an in-app notification informing them of the number of points added.
Alternative Flows	<p>Alternative Flow (No Recorded Information)</p> <ol style="list-style-type: none"> 4a. If no information is recorded in step 2, the app does not proceed with point calculation. 5a. The app does not add any points to the user’s total. 6a. The User does not receive an in-app notification.

Postconditions	<ul style="list-style-type: none"> • Appropriate number of points have been added to the user's total. • The process will restart and continue running in the background.
Scenarios	<ol style="list-style-type: none"> 1) A User logs in to the app, and it begins tracking their exercise. They then walk to the shops and back, and reopen the app once they get home. The app detects the distance they walked and awards them an appropriate number of points. 2) A User logs in to the app as in Scenario 1 and then drives to the shops. The distance is recorded, but the information from the accelerometer and pedometer determines that they were not walking or cycling. Therefore, unlike in Scenario 1, they will not be awarded points for this distance.

Use Case: Open Sticker Pack	
Actors	Primary Actor: User - Initiates the process by selecting 'Open Sticker Pack' from the main menu, and chooses a sticker pack to open.
Preconditions	<ul style="list-style-type: none"> • The user is logged into the app • The user has selected a language to use
Flow of Events	<ol style="list-style-type: none"> 1. The use case starts when the user selects "open sticker pack" from the main menu. 2. The system asks the user which pack they would like to open which can be selected from a list of options 3. The system then checks that the user has sufficient points to be able to open the pack that was selected 4. The system once it has made sure that the user has sufficient points will subtract the amount of points in which it used to open the sticker pack away from the user's points 5. The system then displays a confirmation message that the sticker pack has been purchased as well as how many points the user has left 6. The system then adds the sticker which was received from opening the pack into the user's sticker album 7. The system then displays the sticker/ stickers which the user received from the pack 8. The system finally asks if the customer would like to open another pack

Alternative Flows	<p>Alternative Flow (Insufficient Points) 3a. If the user does not have sufficient points to open the pack selected an error message will be displayed and they will be redirected back to the page in which they can select a different sticker pack to open</p> <p>Alternative Flow (Sticker not unlocked) 6a. If the user does not already have the sticker received in the pack it must be unlocked</p> <p>Alternative Flow (Sticker already in album) 6b. If the user already has the sticker in their sticker album then the user will receive a duplicate (which it can trade at a later date)</p> <p>Alternative Flow (User wants to open another pack) 8a. If the user selects “Yes” then the user will be returned to the different pack options</p> <p>Alternative Flow (User does not want to open another pack) 8b. If the user selects “No” then the user will return to the main menu</p>
Postconditions	<ul style="list-style-type: none"> • The system has updated the user’s sticker album based on the sticker/ stickers which it received from opening sticker packs • The system has updated the amount of points in which the user has based on how much and how many packs were opened • The system (once the user has selected “No”) will return to the main menu
Scenarios	<p>Scenario 1: The user has amassed points either through walking, cycling, or scanning public transport tickets. The user decides to use these points and selects “open sticker pack” from the main menu, they are then given a list of options as to which pack they would like to open, each with its own point cost and probabilities for them to contain different qualities of sticker. Once the user decides which to open, the system checks their point total to see if they can afford the pack. The user has enough points to purchase the pack so their point total is updated. A message is displayed confirming their purchase and showing their remaining point total. The user then opens the pack and receives whichever sticker it contains. Finally, the user is asked whether they would like to open another pack to which they say no. The user is then returned to the main menu.</p>

	<p>Scenario 2: The user chooses the option to “open sticker pack” from the main menu. The customer is then requested to select which pack they would like to open based on the list of options. The user then decides to open the pack which costs X amount of points to open. The system then checks to see if the user has a sufficient amount of points to be able to open the pack which was selected. However the user does not have X amount of points available to purchase the selected sticker pack, therefore an error message will be displayed stating “Not enough points”. The user is then asked whether they would like to open another pack (“Yes”) or if they would like to return to the main menu(“No”). The user selected “No” and the system returned to the main menu.</p>
--	--

Use Case: Connect to Social Media	
Actors	<p>Primary Actor: User - Interacts with the app, connecting it to their social media account. Also provides necessary credentials and permissions</p> <p>Secondary Actor: The social media platform is an external actor that receives connection requests from the app and authenticates the user, who provides the necessary permissions to the app.</p>
Preconditions	<ul style="list-style-type: none"> • User is logged into the app • User desires to connect the app to their chosen social media platform
Flow of Events	<ol style="list-style-type: none"> 1. The user initiates the connection by selecting the "Connect to Social Media" option. 2. The system presents a list of social media app options. 3. The user selects the desired social media platform. 4. The system initiates the authorization process, connecting the user to the chosen social media platform. 5. The system redirects the user to the social media login page of the selected app. 6. The user logs in to the social media app and grants the necessary permissions. 7. The system retrieves relevant data, such as profile information, friends, and followers, based on the granted permissions. 8. The system notifies the user of a successful connection and requests confirmation to access and display social media data within the app.
Alternative Flows	<p>Alternative Flow (User Denies Authorisation):</p>

	<p>6a. If the user denies authorisation or does not grant necessary permissions:</p> <p>7a. The system displays a notification or message indicating the denial.</p> <p>8a. The system does not retrieve social media data.</p> <p>Alternative Flow (Authorisation Error):</p> <p>6b. If there is an error during the authorisation process:</p> <p>7b. The system displays an error message indicating the issue.</p> <p>8b. Provides options for the user to retry the connection or cancel.</p>
Postconditions	<ul style="list-style-type: none"> • The user's app account must be successfully connected to the chosen social media platform • Relevant social media data is retrieved and stored for app integration • The user must receive confirmation of a successful connection.
Scenarios	<p>The actor in this case is the user.</p> <p>Scenario: Sharing Achievements with stickers</p> <p>A user is a frequent user of a mobile app that promotes sustainability through different sustainable modes of transport, for example, trains and buses, as well as physical activity that further promotes sustainability, such as walking, running and cycling.</p> <p>The app integrates a "connect to social media" feature which allows users to share their achievements, which in this app are in the form of virtual images (stickers) which can be earned by taking sustainable forms of transport/physical activity.</p> <p>The user's goal of using the app is to track their eco-friendly and sustainable commutes/activities. The user can achieve this by connecting the app to showcase their progress, sharing the stickers as a result of adopting sustainable modes of transport and engaging in more physical activities.</p> <p><u>Scenario 1 - Successful</u></p> <ol style="list-style-type: none"> 1. App login - the user opens the app, logs into their account, and navigates through to the "view stickers" option 2. Exploring Sticker Collection - on the "view stickers" screen, the user wishes to view their sticker collection, including unique virtual images based on the type of activity completed during the week. 3. Percentage Completion - The user then checks their percentage completion, showing how many stickers have been collected out of a possible maximum of different types of stickers

4. Choosing Social Media connection - the user then decides to share their achievements on their chosen platform, in this scenario, Instagram. The user selects the “Connect to Social Media” option on the app
5. Authentication Process - The system detects that the user wishes to log on and initiates the authorisation process, which redirects the user to the Instagram login page. The user enters their Instagram credentials.
6. Granting permissions - Instagram prompts the user to grant necessary permissions for the app, for example, social media posts/data
7. Captions and Sticker Showcase - The user has the option to compose a personalised caption for their Instagram post.
8. Confirmation and completion - The user receives a confirmation in the app that their account is successfully connected to Instagram. The app ensures that this connection complies with Instagram’s policies.
9. Sharing Sticker Achievements - The user selects a recent walking achievement/milestone from their sticker collection. They compose a caption and confirm the sticker selected. They then share their sustainable achievement on Instagram.
10. Widespread engagement - The user’s Instagram followers engage and interact with the post. When viewing the virtual stickers that the user posted, the widespread public express interest in adopting more sustainable commuting and exercise habits for themselves.

Outcome:

The user successfully connects their app to their Instagram profile, sharing their sustainable achievements through unique stickers.

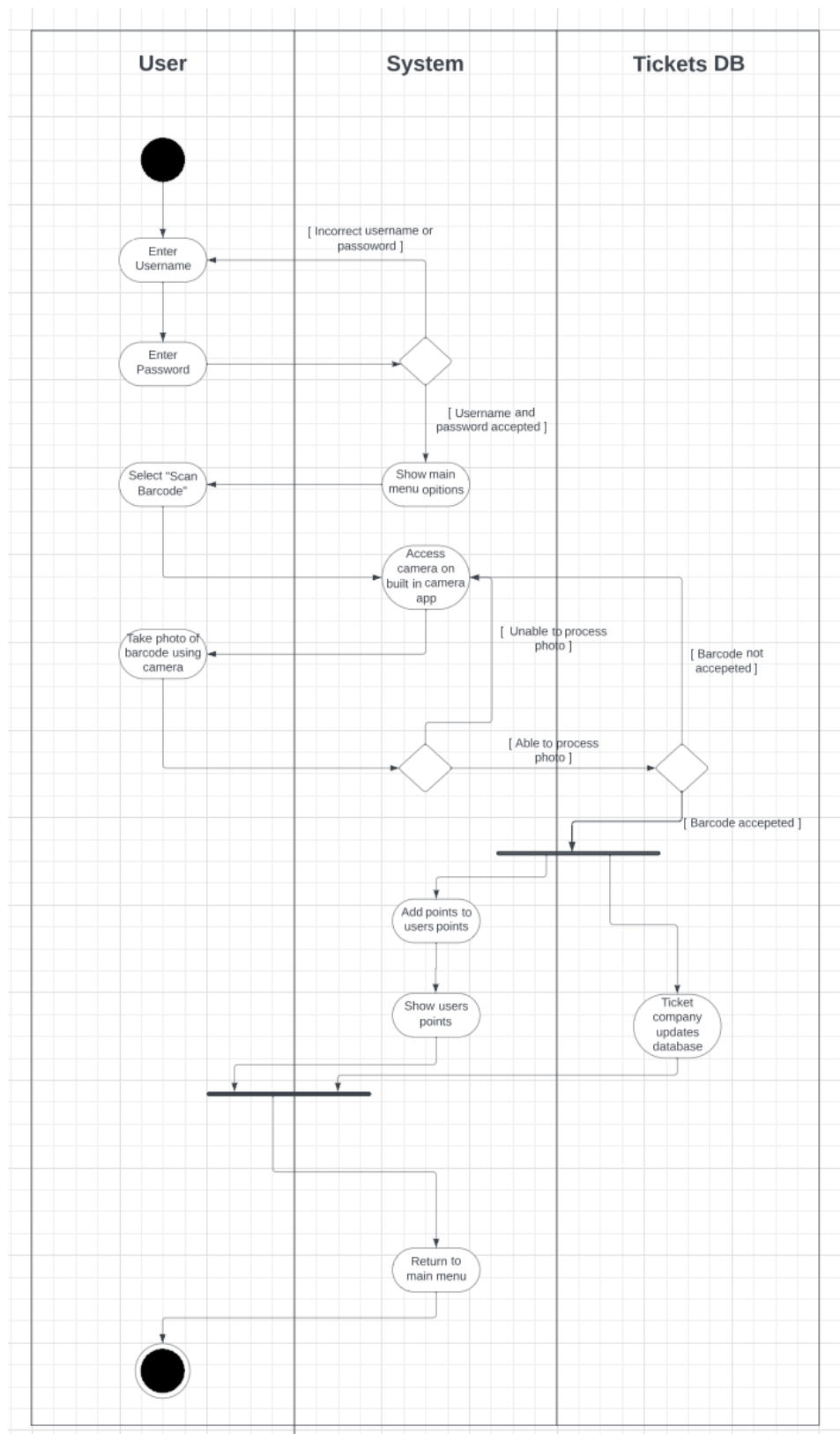
Scenario 2 - Unsuccessful and Resolved

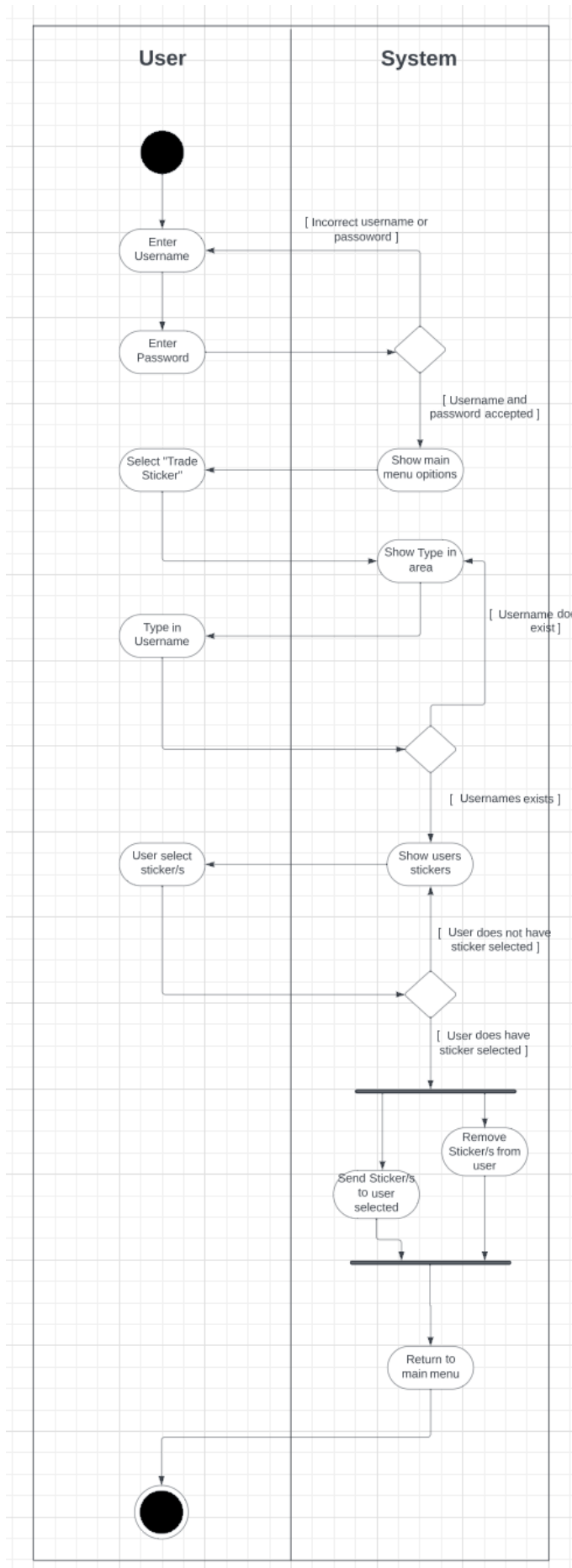
A user has earned stickers by taking eco-friendly journeys. The user attempts to connect the app to Facebook, however, they face a problem with the app.

The user initiates the process to connect the app to Facebook to share achievements, however, a glitch occurs during the authorisation process, which did not lead to a successful connection. The app detects the error and provides a simple error message. The message displayed to the user explains that there has not been a successful connection and that this might be down to a poor/unstable internet connection or other issue. The user has the option of selecting one of two options: “Retry” or “ Contact Support”.

	<p>The user selected “Contact Support” because there did not seem to be an issue with their internet. Once the detailed problem was submitted by the user, the app kept the user informed about the ongoing investigation of the issue.</p> <p>The app notified the user about what steps the user can take to solve the issue temporarily, before providing an update later that resolves the issue.</p>
--	---

B4: Activity Diagrams





B5: Class Analysis

Noun/verb analysis:

Noun/verb analysis has been applied to all points listed in the requirements section to identify the candidate classes for our system, as well as their operations. The following table of nouns were each identified as a potential class.

Candidate class	Use	Candidate class	Use
Touchscreen	Attribute (IOhandler)	Distance walked	Attribute (DistanceTracker)
Keyboard input	Attribute (IOhandler)	Distance cycled	Attribute (DistanceTracker)
Camera input	Attribute (IOhandler)	Distance by bus	Attribute (Ticket)
User	Class	Distance by tram	Attribute (Ticket)
GPS receiver	Attribute (IOhandler)	Distance by train	Attribute (Ticket)
Points	Attribute (UserData)	Sticker	Class
Accelerometer	Attribute (IOhandler)	Sticker store	Class
Audio alerts	Attribute (Notifications)	Sticker pack	Attribute (StickerStore)
Notifications	Class	Sticker album	Attribute (UserData)
Language	Attribute (UserData)	Percent completed	Attribute (UserData)
Ticket scanning	Class	Sticker trading	Class
Bus ticket	Class (generic Ticket)	Username	Attribute (UserData)
Train ticket	Class (generic Ticket)	Password	Attribute (UserData)
Tram ticket	Class (generic Ticket)	Stickers to send	Attribute (StickerTrading)
Ticket ID	Attribute (Ticket)	Stickers sent back	Attribute (StickerTrading)
Distance tracking	Class		

The following table of verbs were suggested as potential operations for these candidate classes:

Verb	Possible class for method	Verb	Possible class for method
Process GPS input	WalkCycleTracker	Calculate public transport distance	WalkCycleTracker
Process accelerometer input	WalkCycleTracker	Grant points	UserData
Send notification	Notifications	Save points	UserServer
Select language	UserData	Buy random sticker	StickerStore
Determine whether walking or cycling	WalkCycleTracker	Buy sticker pack	StickerStore
Calculate walking distance	WalkCycleTracker	Buy fresh sticker pack	StickerStore
Calculate cycling distance	WalkCycleTracker	Save stickers	UserServer
Scan bus ticket	TicketScanner	Display stickers	UserData
Scan bus pass	TicketScanner	Trade stickers	StickerTrading
Scan train ticket	TicketScanner	Select stickers to trade	StickerTrading
Scan train ticket	TicketScanner	Send stickers to trade	StickerTrading
Scan tram ticket	TicketScanner	Add sticker to collection	UserData
Scan tram pass	TicketScanner	Share completion on Instagram	UserData
Verify bus ticket code	TicketScanner	Share completion on Twitter	UserData
Verify train ticket code	TicketScanner	Share completion on Facebook	UserData
Verify tram ticket code	TicketScanner	Share completion on WhatsApp	UserData

Responsibility analysis using CRC cards:

Following on from the noun/verb analysis, we created CRC (Class, responsibilities and collaborations) cards for each candidate class, which help to explain how these classes would work with one another within the overall system. The responsibilities of a class provide a brief outline of the purpose of that class, while the collaborators are the classes it will need to directly interact with in order to function.

UserServer	
Responsibilities	Collaborators
Maintain a record of all key info on all users that have created an account with the app. Used for cloud storage of users' progress, in addition to searching for other users to trade stickers with online.	User data StickerTrading

UserData	
Responsibilities	Collaborators
Maintain data concerning a particular user of the app, namely their current points and sticker album progress. Periodically upload all data stored locally on the device to the user server.	UserServer PointsCalculator Notifications StickerStore

TicketScanner	
Responsibilities	Collaborators
Scan and validate legitimate, unused public transport tickets by opening the user's camera and prompting them to scan a barcode or QR code and checking the relevant ticket database.	Ticket

WalkCycleTracker	
Responsibilities	Collaborators
Determine whether the user is walking or cycling based on the device's accelerometer/GPS receiver and track how far they travel using that mode of transport.	PointsCalculator

Sticker	
Responsibilities	Collaborators
Maintain data relating to a particular sticker, namely the image used, the name, and the amount of this sticker owned by the user.	StickerPack

StickerPack	
Responsibilities	Collaborators
Maintain data relating to a pack of stickers that can be purchased from the sticker store. This includes the price of the pack, as well as its type (whether it is a normal pack or a fresh pack).	Sticker StickerStore

PointsCalculator	
Responsibilities	Collaborators
Calculate how many points to grant to a user based on a given activity.	UserData Ticket WalkCycleTracker

Ulmanager	
Responsibilities	Collaborators
Maintain a collection of menu screens and allow the user to move between them with touchscreen or keyboard input.	Notifications

IOhandler	
Responsibilities	Collaborators
Maintain a collection of relevant input/output controllers of the user's device for the purpose of navigating the app's interface and reading the accelerometer/GPS receiver.	Ulmanager Notifications

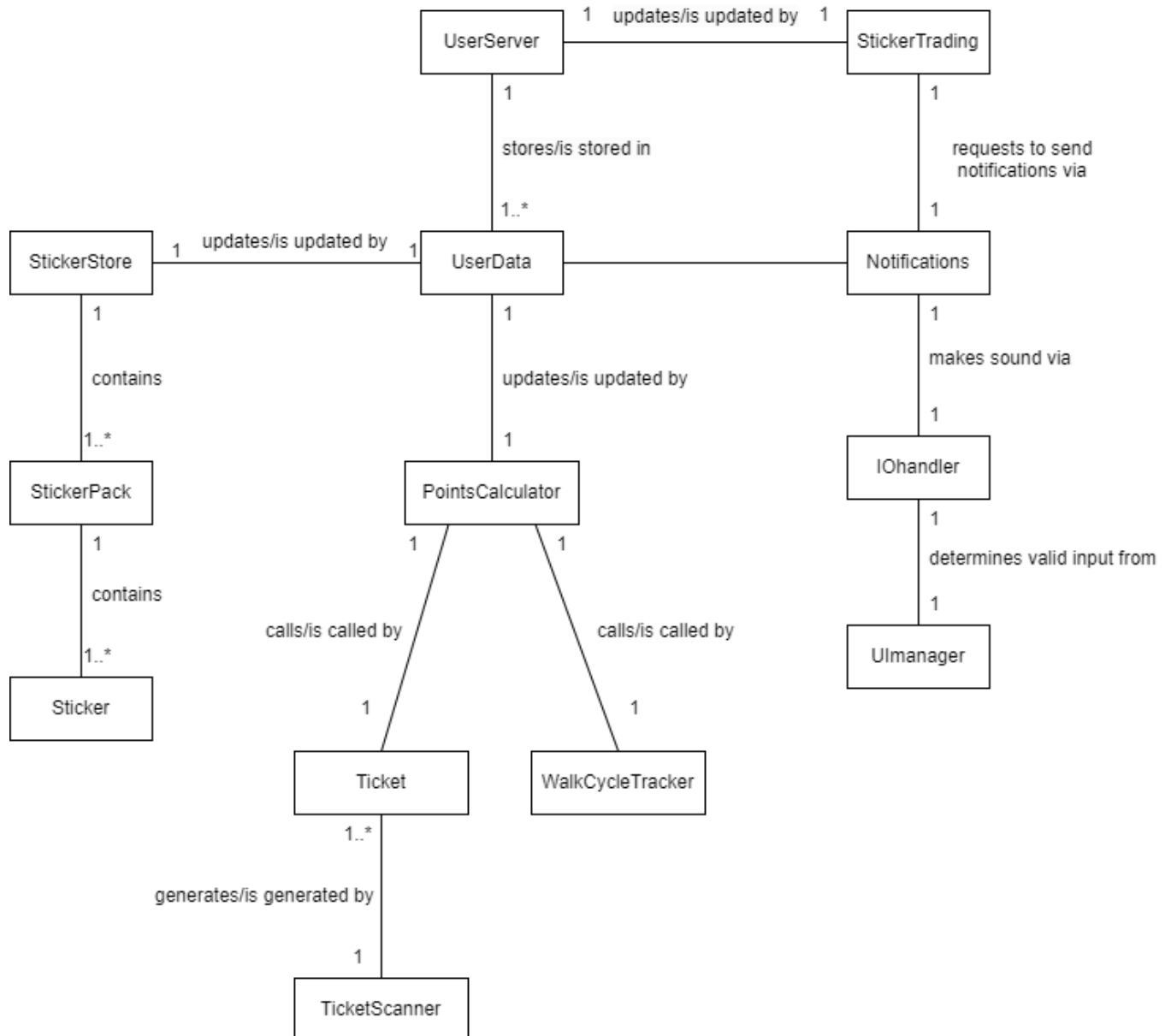
StickerStore	
Responsibilities	Collaborators
Maintain a store where the user can spend their points to buy a single random sticker, a pack of random stickers or a fresh pack of random stickers guaranteed to contain at least one uncollected sticker. The store then adds the purchased stickers to the user's album in the user data class.	StickerPack UserData

StickerTrading	
Responsibilities	Collaborators
Allow a user to select any number of their owned stickers and request to trade them. Allow another user to select any number of their owned stickers in return and ask to confirm the trade. Allow the two users to confirm a trade and update the sticker album of each user in the user server accordingly.	Notifications UserServer

Notifications	
Responsibilities	Collaborators
Send notifications to the user's device relating to trade requests, as well as reminders to open the app if they have not done so for x consecutive days.	UserData IOhandler StickerTrading

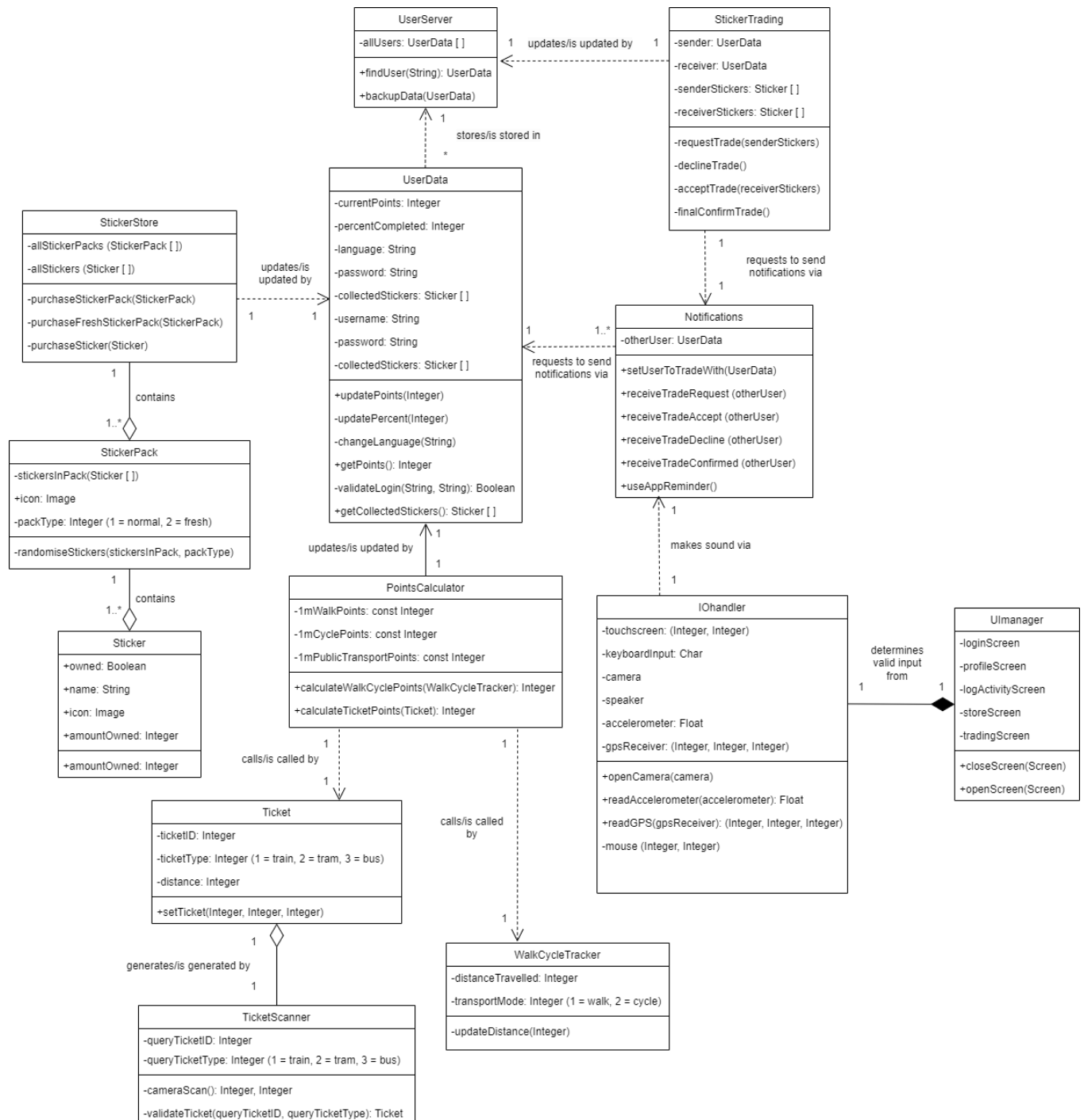
First cut diagram:

The noun/verb analysis was used in combination with the CRC cards to create a first cut class diagram (found below), which outlines the names of the classes and how they relate to each other. The directionality of the class relationships, as well as their multiplicity, is not specified at this point in the class analysis.



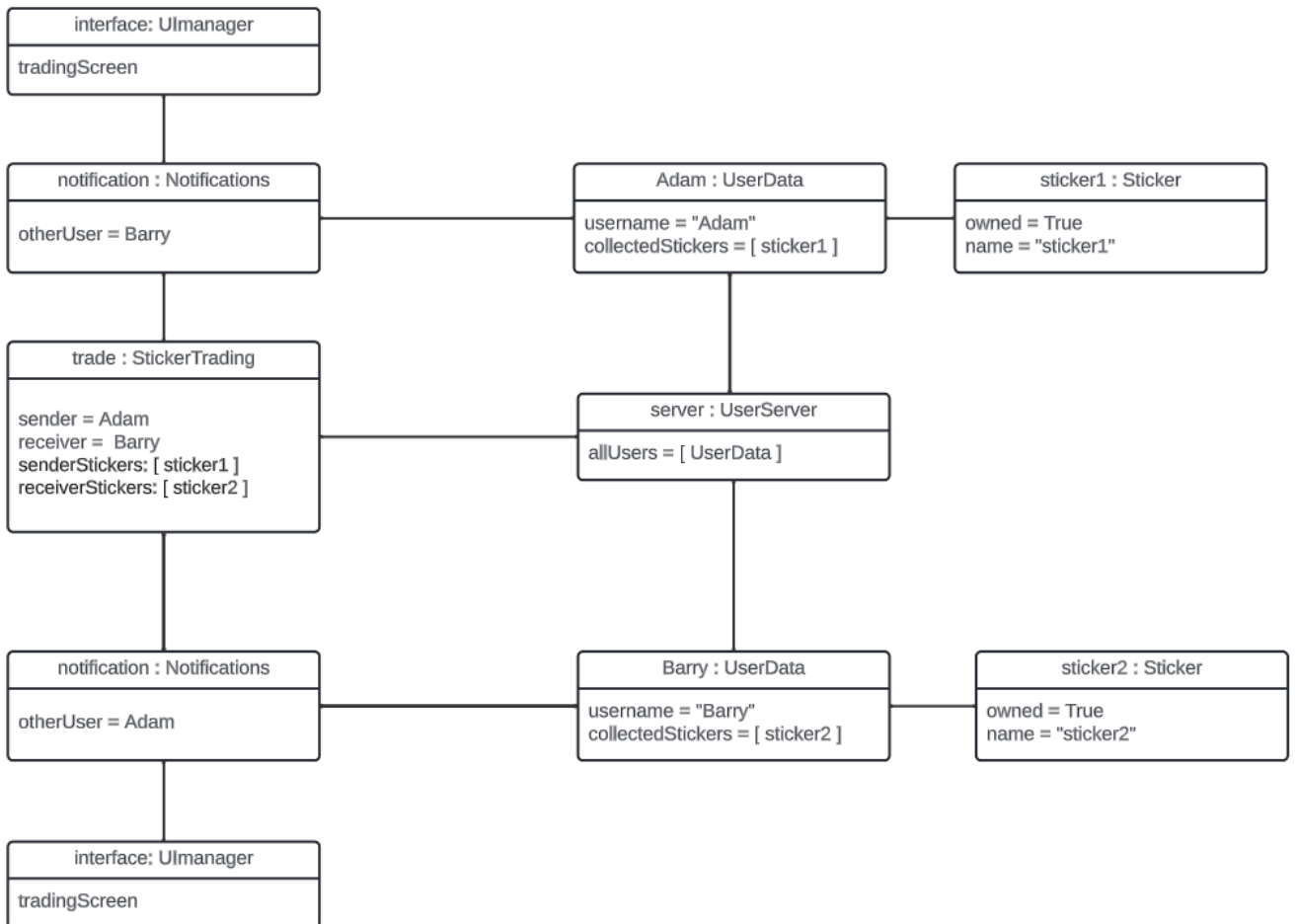
Full class diagram:

From the first cut diagram, the final class diagram was created, which details all the main classes, as well as their relationships, attributes and methods.

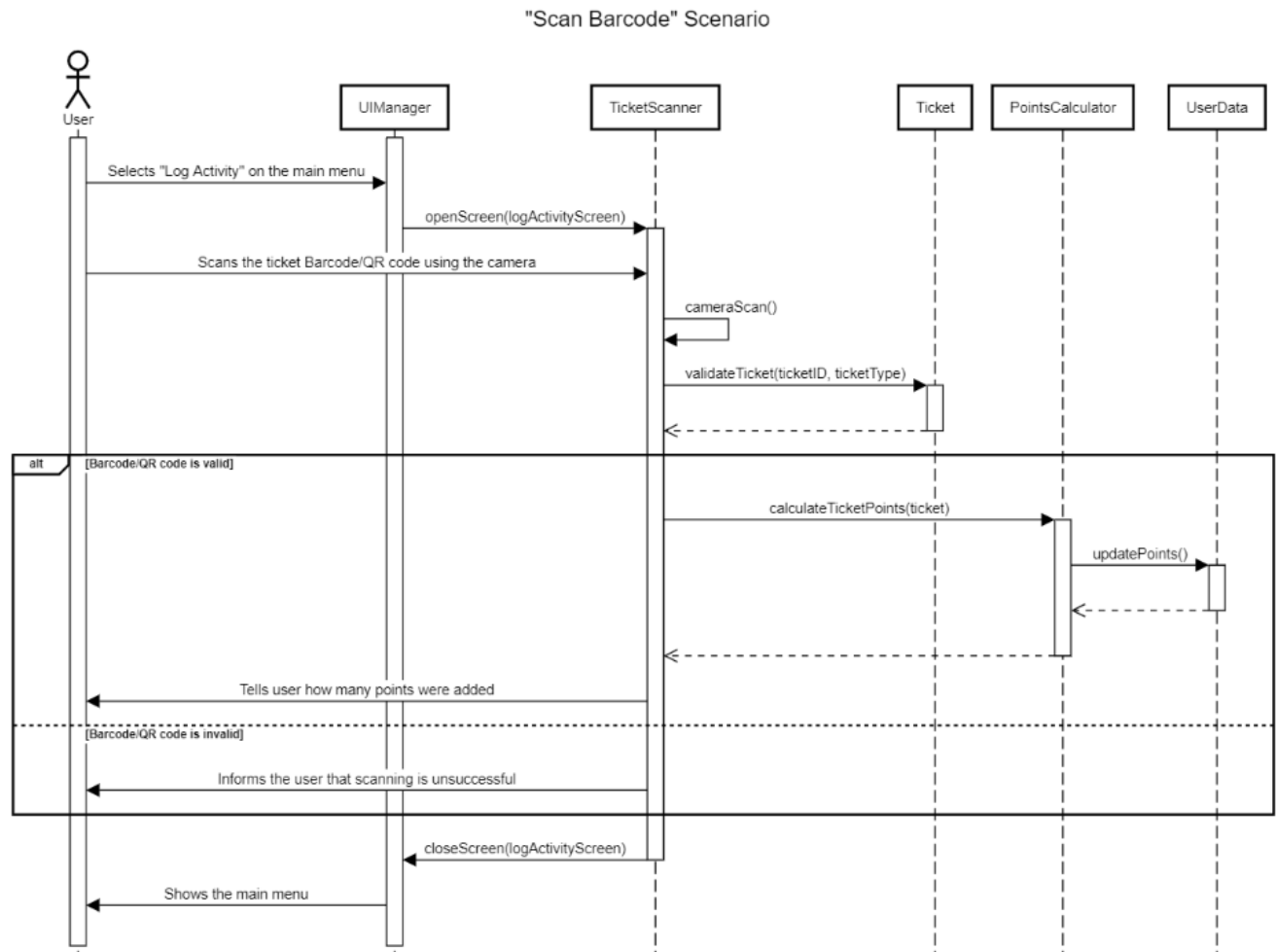


B6: Object Diagram

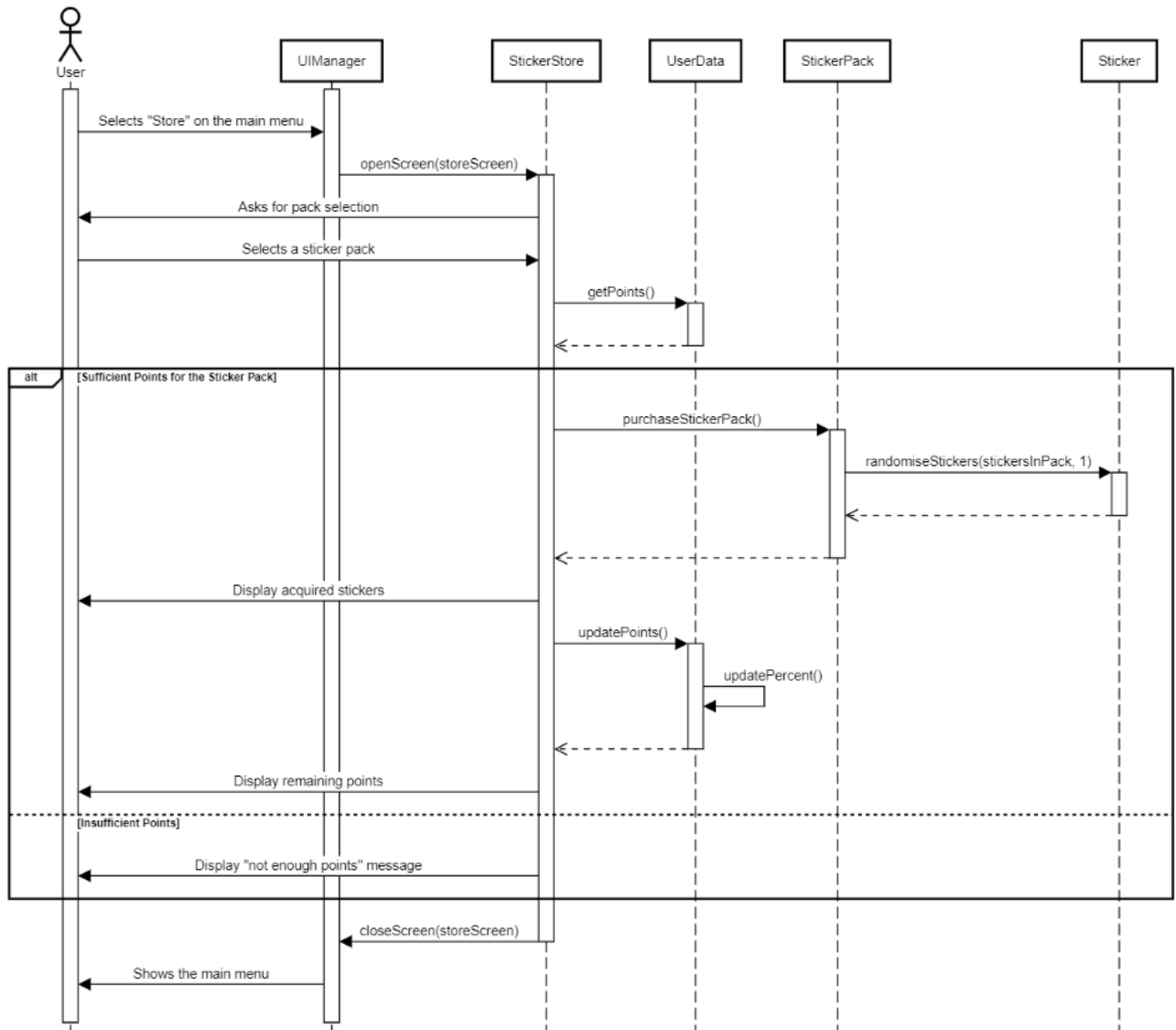
This Object Diagram represents the key objects in a “Trade Stickers” scenario, where the trade is taking place between two users “Adam”, who is initiating the trade and sending “Sticker 1”, and “Barry”, who is sending “Sticker 2”. At the moment captured in the diagram, both users have accepted the trade and selected which stickers to send.



B7: Sequence Diagrams

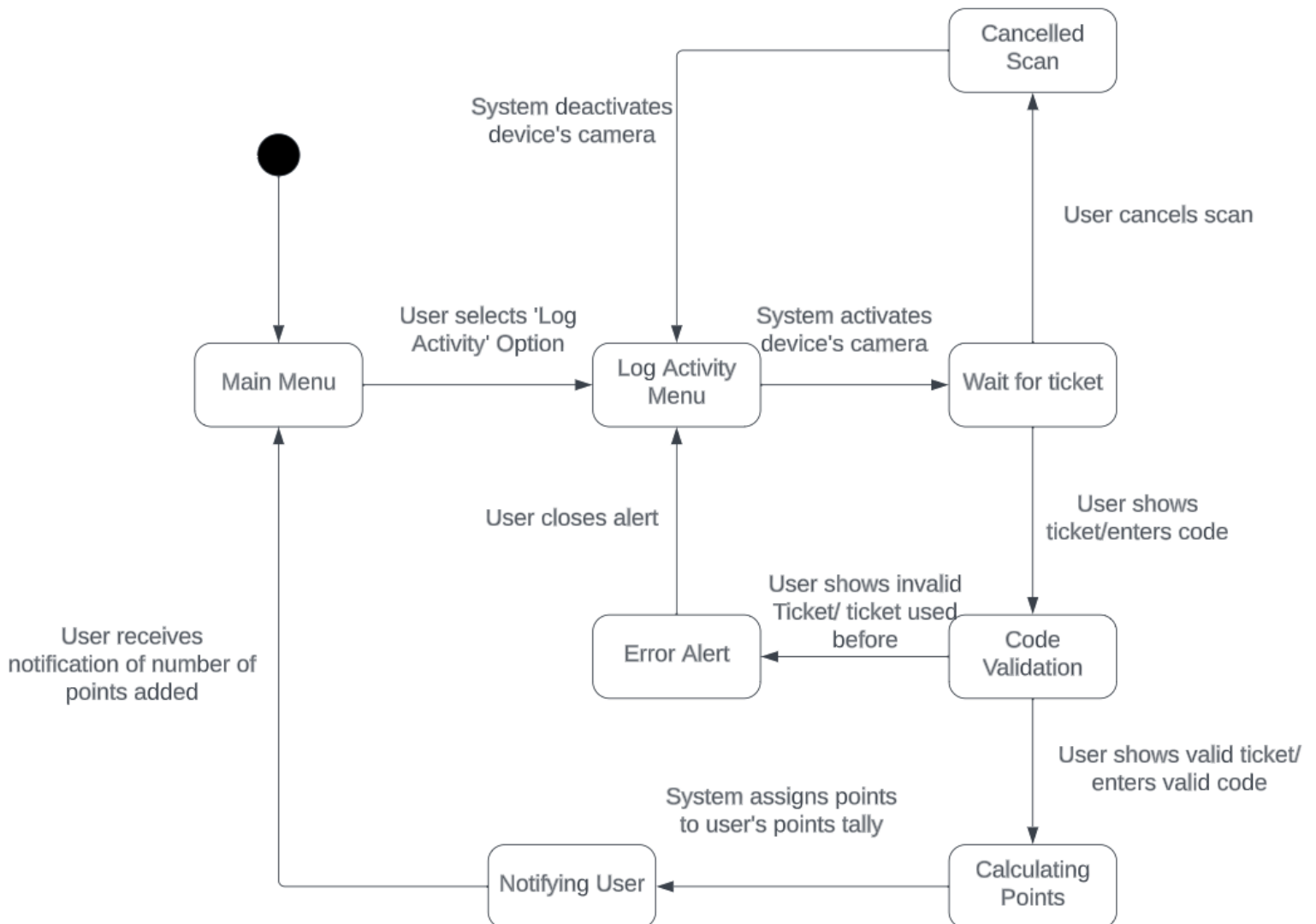


"Open Sticker Pack" Scenario

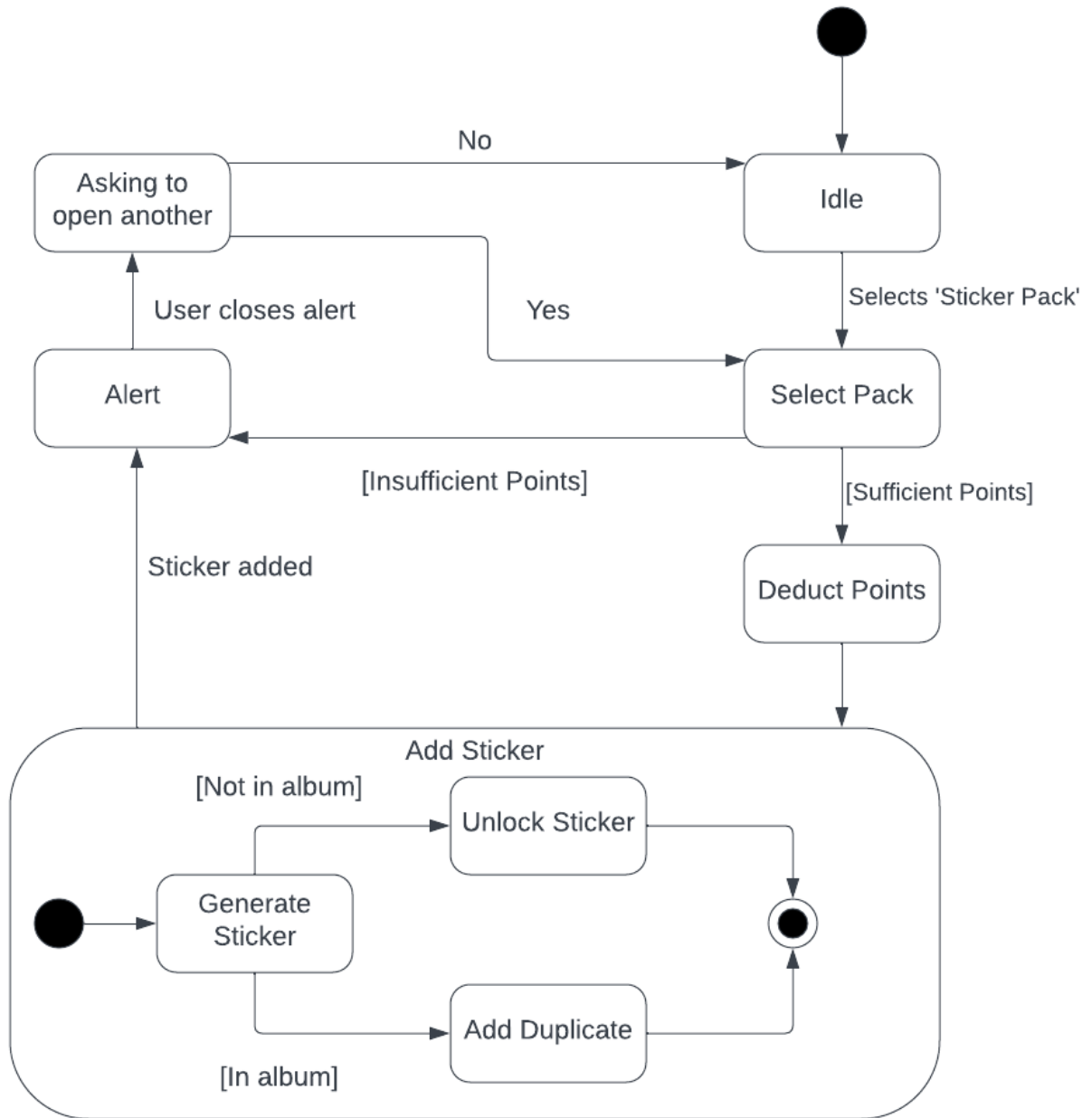


B8: State Machine Diagrams

Scan Barcode State Machine Diagram



Open sticker pack state machine diagram



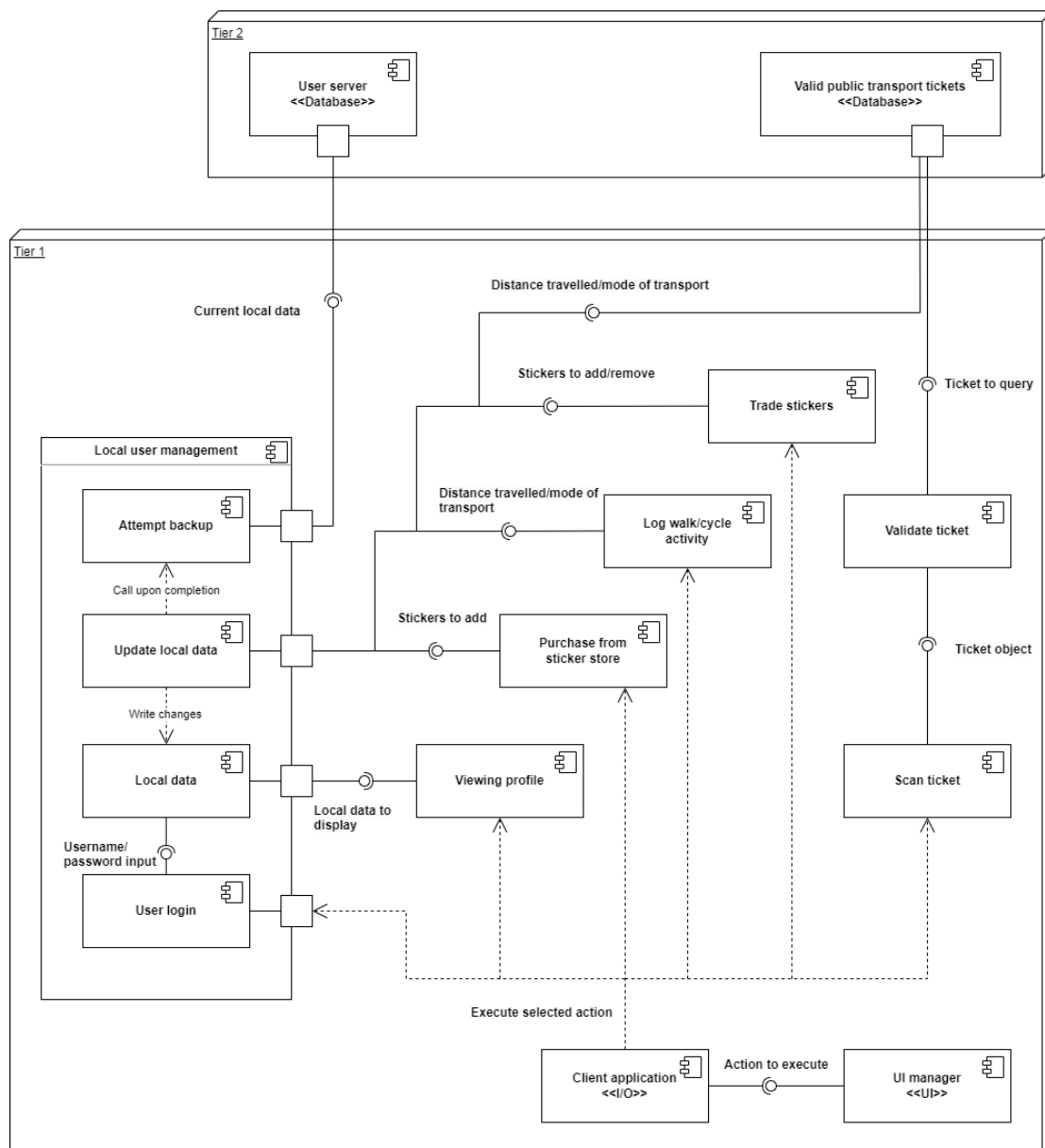
C: Software Architecture Style, Modelling and Evaluation

Two possible designs were proposed for the overall system. Both designs are explained via component and deployment diagrams, followed by a trade-off analysis of each design and a conclusion on our chosen design.

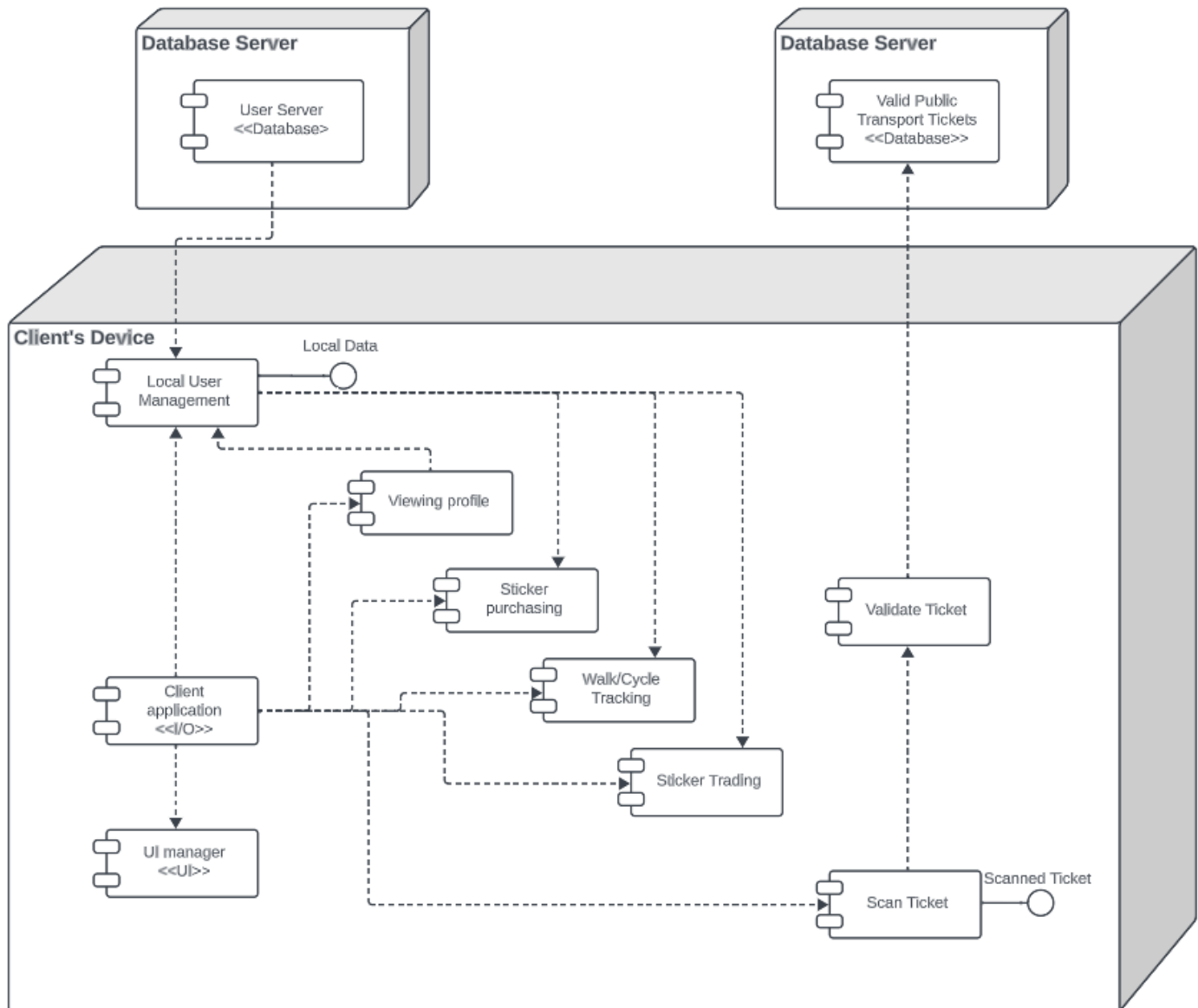
Design one: Two-tier client/server

Component diagram:

The following diagram shows the componentization of a proposed two-tier client/server architecture for the system, including the key interfaces required for the user to interact with it.



Deployment diagram:



Tradeoff analysis

The non-functional requirements were considered when designing the architecture for the system used by the app. Namely, the system should allow the user to interact with many parts of the app, such as tracking walking and cycling distance, while not connected to the internet.

Other considerations include the need for careful storage of the user's data and efficient use of the device's memory to ensure the app feels responsive.

Component diagram explanation:

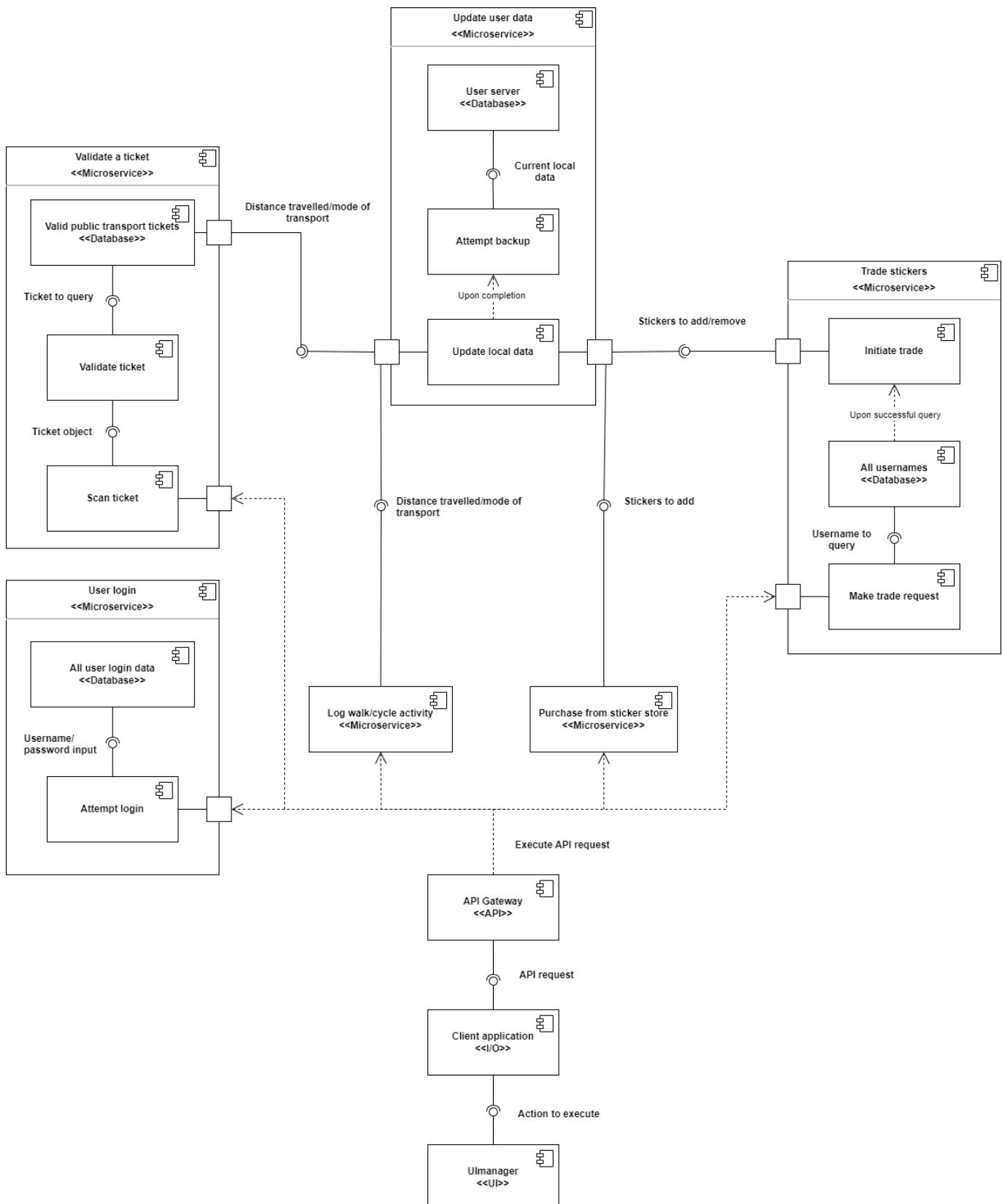
When choosing how to design a two-tier client/server architecture for the system, the main concern was to reduce coupling between the different components in order to maximise the efficiency of the system. For instance, all local data for the given user, as well as the operations performed on said data, are stored as sub-components in a single component named 'Local user management'. This means that only a single component of the client system stores all the local data relating to that client (i.e. user). This is useful when attempting to back up the local data to the user server, as the server does not need to be coupled to many different sub-components for login data, sticker data etc. This also means that updating several aspects of the user's data simultaneously, such as subtracting points and adding stickers to their collection, only requires coupling with one component. The components are also organised in such a way that most of the features of the app can be used without having access to the internet, which will be a common scenario if the user lives in a remote area or is using public transport with inconsistent Wi-Fi connection, such as trains. The user can view their profile, purchase stickers and log their walking and cycling activity while offline. The only features requiring a network connection are trading with other users and backing up the local data to the server. A single database component is used to store all valid public transport tickets, which means it can be queried for bus, train and tram tickets that the user may scan into the app. This means that the 'Scan ticket' component does not need to be coupled with three different databases. This focus on offline usability also ensures that the user's location data, used in the 'Walk/cycle tracking' component, is not coupled to an online server of any kind, such that a server-side security breach would not compromise the user's privacy.

Deployment diagram explanation:

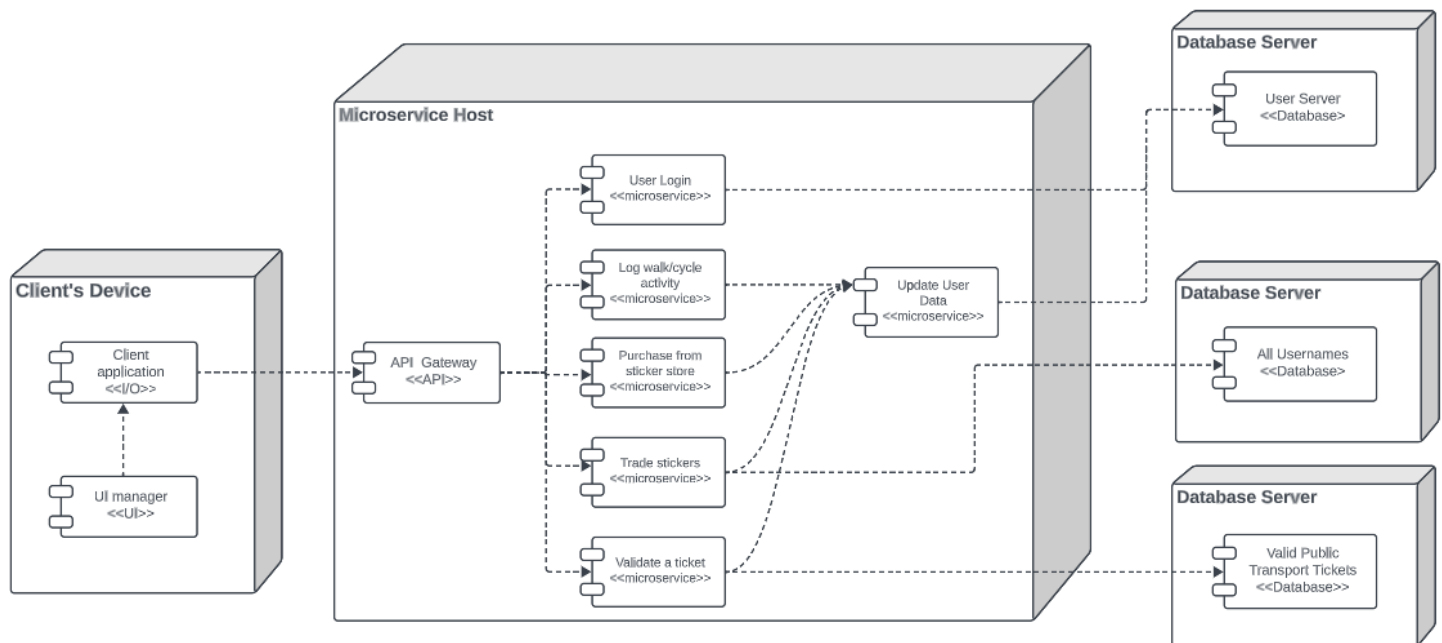
The majority of the components are deployed on the client's mobile device via the installed application, the exceptions being the servers, which are managed remotely. Having most processes running locally reduces the general response time of the application, but has the drawback of increasing the workload on the client's device. While the application is unlikely to be highly resource-intensive, this could cause performance issues for lower-end devices.

Design two: Microservices

Component diagram:



Deployment diagram:



Tradeoff analysis

Component diagram explanation:

The main design difference with our second proposed architecture is the use of separate databases for logging in and trading stickers. Rather than store all the data for all users in a single location, the login data (username and password) and data required to find a user to trade with (i.e. just their username) are stored redundantly within their own microservice components. This decentralised approach means that each microservice can be independently deployed and worked on (for the most part), which improves the maintainability of the system. If a single microservice is found to not be working as intended, it can be taken down temporarily and worked on without impacting the rest of the system. The downside is that storing redundant data takes up more storage, both on the user side in the case of login data, and on the service side in the case of the usernames database. Another consideration is that many of the microservices must still be coupled to a single 'Update User Data' component to avoid data synchronisation issues. There is also an added security risk, due to how the user's location data must be sent to the microservice containing the user server. This would need to be addressed by implementing high-grade encryption of the data, lengthening development time. The microservices architecture does have some security benefits, however; the client never has direct access to the user server and this reduces the risk of a malicious user obtaining sensitive information on another user (e.g. their login details).

Deployment diagram explanation:

In this strategy, the minimum amount of components are deployed onto the client's device. The rest of the components are hosted remotely as microservices. These microservices would require additional server space to be hosted on, increasing the cost of the project. However, the file size of the client application would be smaller, and the reduced complexity (only 2 components) means ensuring functionality on different devices would be simpler.

Final Choice of design:

When evaluating which of the proposed architecture styles to choose for the system, we must analyse and compare the trade-offs of each style.

The first proposed style is a two-tier client-server architecture. All processes take place on the client's device. User data is backed up to a remote server whenever changes occur, and data is retrieved from the server whenever a user logs in. The system may also access a server to check validity of scanned tickets. The benefits this architecture provides include:

- + Allowing the user to access most services of the app without connecting to the internet.
- + Processes happening locally reduce the app's response time.
- + Improved security due to sensitive information such as location and travel data only being held locally.
- + Simplicity of development: most components are contained within a single application.

However, there are also trade-offs associated with this architecture:

- Any updates to the app's functions will require the user to update the app on their system for the changes to be implemented.

The second proposed style is a microservices architecture, where the client application communicates with remotely hosted microservice processes via an API gateway. The microservices in turn are connected to servers for storing user data and checking ticket validity. The benefits of this architecture include:

- + The microservices being remotely hosted and disconnected from one another means they can be modified individually without requiring updates to the client application.
- + Clients do not have direct access to the servers, improving security.

Conversely, this architecture's trade-offs:

- Requires users to be connected to the internet to use the app's services.
- Sensitive information such as location data will be transmitted over the internet; caution will have to be taken to ensure these transfers are secure or this could lead to security risks.
- Increased development complexity compared to the two-tier architecture, due to more "moving parts".

Based on analysis of the respective benefits and trade-offs of each of these architectures, it was decided that the system should be implemented with the two-tier client-server architecture. When comparing security benefits, client-server holding the most sensitive user data locally is preferable to microservices requiring it to be shared. Client-server does have the liability of a direct connection between the client and server, but if suitable precautions are taken to ensure connections are secure then this trade-off is acceptable. Furthermore, the ability for users to interact with the app whilst offline is a desirable feature for ease of use.

D: Testing

1. Introduction

A mobile application designed to reward users for using environmentally sustainable modes of transport such as walking, cycling, and public transport. The goal is to promote behavioural change by reducing dependence on environmentally damaging modes of transport, accelerating progress towards net zero. Within the system, users can gain points either proportional to their distance travelled by walking / cycling or by entering public transport tickets.

Project Name: Environmental Transport App

Document Owner: [Group 63]

Document Version: 1.0

Date: November 1st, 2023

2. Testing Objectives

- To ensure the functionality of the system
- To verify that the system meets the core requirements
- To verify that the system does not overbear the host device's capabilities

3. Features To Be Tested

- Feature 1: Language Selection - verify that users can select and switch between different languages, and ensure that the app's interface and notifications can adapt to the changes
- Feature 2: Detecting distance travelled via walking & gaining points for it - test for the accuracy of the distance detection algorithm that calculates points. Ensure that users are consistently earning points corresponding to their physical activity.
- Feature 3: Inputting a train ticket via the device's camera & gaining points for it - Evaluate the functionality of the ticket scanning feature and ensure that the camera accurately captures ticket details, and users are appropriately rewarded with points.
- Feature 4: Inputting a bus ticket via a code & gaining points for it - Validate the effectiveness of manual ticket input, again ensuring that users will be appropriately rewarded with points.
- Feature 5: Trading points for stickers & losing the points - Ensure that points to sticker trade is executed correctly by checking that users lose the correct amount of points when trading and stickers are correctly added to the collection.
- Feature 6: Viewing collected stickers in the gallery - Test the user's ability to view collected stickers, ensuring that the gallery displays stickers correctly.
- Feature 7: Trading stickers with other users - Confirm that users can successfully initiate and complete sticker trades with other users.
- Feature 8: The system uses less than 20% of a device's processing power while running in the background
- Feature 9: The system occupies less than 10% of a device's memory while running in the background

- Feature 10: The system can interact with the same system on a different mobile operating system - test that app's ability to interact seamlessly between different platforms.

4. Test Strategy

We will make use of the following types of testing:

- Unit testing: To verify the correctness of individual system functions
- Integration testing: To test the interactions between different components
- System testing: To evaluate the overall system and validate its performance
- Acceptance testing: To determine if the system meets the given requirements

5. Test Cases

Please see the table below (on the next page)

6. Schedule

Test Preparation: November 1st, 2023 - 19th November, 2023

Testing Execution: 20th November, 2023 - 26th November, 2023

Defect Reporting: 27th November, 2023 - 30th November, 2023

Test Summary Report: 5th December, 2023

7. Test Completion Criteria

All defects relating to M-priority or S-priority requirements are resolved.

All other defects are resolved or deferred.



Test coverage meets predefined criteria.




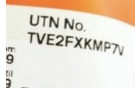
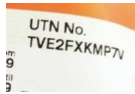

8. Assumptions:

N/A

9. Appendix

N/A

Test Case ID	Test Scenario	Test Case Title	Prerequisites	Test Steps	Test Data	Expected Result (ER)	Actual Result	Priority	Result	Comments
TC_SL_001	(TS_001) Select Language	Verify navigating to "Select Language" page from "Settings" page	Open the app [what the app is called] and navigate to the "Settings" page	1 Click on "Select Language" from "Settings" page (Verify ER - 1)	Not applicable	1 User should be navigated to "Select Language" page				
TC_SL_002	(TS_001) Select Language	Verify selecting the drop down box on the "Select Language" page	Open the app [what the app is called] and navigate to the "Settings" page	1 Click on drop down box on "Select Language" page (Verify ER - 1) 2 Click on the drop down box on the "Select Language" page (Verify ER - 2)	Not applicable	1 User should be navigated to "Select Language" page 2 User should be shown the list of available languages in the drop down box				
TC_SL_003	(TS_001) Select Language	Verify selecting a language on the "Select Language" page	Open the app [what the app is called] and navigate to the "Settings" page	1 Click on drop down box on "Select Language" page (Verify ER - 1) 2 Click on the drop down box on the "Select Language" page (Verify ER - 2) 3 Select a language from the options in the drop down box <Refer Test Data> (Verify ER - 3)	Select Language : Urdu	1 User should be navigated to "Select Language" page 2 User should be shown the list of available languages in the drop down box 3 Language of the app should change to Urdu				
TC_RA_001	(TS_002) Record Activity	Verify detecting the user walking	Open the app [what the app is called] to the 'Main Menu' page	1. Walk X amount of metres <Refer Test Data>	1 User will walk X amount of metres	1: 'Points' on the main menu should be updated to the users new total based on the distance detected walking				
TC_RA_002	(TS_002) Record Activity	Verify points are calculated based on how much the user has walked	Open the app [what the app is called] to the 'Main Menu' page	1. Walk X amount of metres <Refer Test Data>	1 User will walk X amount of metres	1. The 'Points' on the main menu should increase by the correct amount based on the distance walked				
TC_RA_003	(TS_002) Record Activity	Verify points are added to the users points	Open the app [what the app is called] to the 'Main Menu' page	1. Walk X amount of metres <Refer Test Data>	1 User will walk X amount of metres	1: 'Points' on the main menu should be updated to the users new total				
TC_SB_001	(TS_003) Scan Barcode	Verify navigating to "Log Activity" screen from "Main Menu" page	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1)	Not applicable	1 User should be navigated to "Log Activity" page				
TC_SB_002	(TS_003) Scan Barcode	Verify scanning a train barcode which is valid to use	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1) 2 Take a picture of a valid barcode using the camera <Refer Test Data> (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that points from barcode have been added to the users points				
TC_SB_003	(TS_003) Scan Barcode	Verify scanning a train barcode which is not valid to use as it cannot be interpreted	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 Barcode (shown in test data) would be blurry on camera	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1) 2 Click on "Scan Ticket" 3 Take a picture of a invalid barcode (one which cannot be interpreted) using the camera <Refer Test Data> (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that the photo could not be interpreted and asks the user to take another photo				

TC_SB_004	(TS_003) Scan Barcode	Verify scanning a train barcode which is not valid as it is not in database	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1) 2 Click on "Scan Ticket" 3 Take a picture of a invalid barcode (one which is not in the database) using the camera <Refer Test Data> (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that the barcode provided is not valid				
TC_SB_005	(TS_003) Scan Barcode	Verify scanning a train barcode which has already been used	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 Barcode (photo in test data) which will be scanned is saved as "already been used"	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1) 2 Click on "Scan Ticket" 3 Take a picture of a invalid barcode (as it has already been used) using the camera <Refer Test Data> (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that the barcode has already been used				
TC_SB_006	(TS_003) Scan Barcode	Verify points from scanning a train barcode are added to the users points	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" page (Verify ER - 1) 2 Click on "Scan Ticket" 3 Take a picture of a valid barcode using the camera <Refer Test Data> (Verify ER - 2) and (Verify ER - 3)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that points from barcode have been added to the users points 3 Users points will increase (the amount is dependant on the barcode)				
TC_E_C_001	(TS_004) Enter Code	Verify navigating to "Log Activity" screen from the "Main Menu" screen	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" screen (Verify ER - 1)	Not applicable	1 User should be navigated to "Log Activity" page				
TC_E_C_002	(TS_004) Enter Code	Verify entering a bus code which is valid to use	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" screen (Verify ER - 1) 2 Enter a code which is valid <Refer Test Data> into the text box provided (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that points from code have been added to users points				
TC_E_C_003	(TS_004) Enter Code	Verify entering a bus code which is not valid as it is not in database	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" screen (Verify ER - 1) 2 Enter a code which is invalid <Refer Test Data> (as it is already in the database) into the text box provided (Verify ER - 2)	T222222 (Random Code not in database)	1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that the code entered is not in the database				
TC_E_C_004	(TS_004) Enter Code	Verify entering a bus code which is not valid as it has already been used	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 Code (TVE2FXKMP7V) which will be used is saved as "already been used"	1 Click on "Log Activity" from "Main Menu" screen (Verify ER - 1) 2 Enter a code which is invalid <Refer Test Data> (as it has already been used) into the text box provided (Verify ER - 2)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that the code entered has already been used				
TC_E_C_005	(TS_004) Enter Code	Verify points are added to user points after entering valid bus code	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Log Activity" from "Main Menu" screen (Verify ER - 1) 2 Enter a code which is valid <Refer Test Data> into the text box provided (Verify ER - 2) and (Verify ER - 3)		1 User should be navigated to "Log Activity" page 2 Message will be displayed stating that points from code have been added to users points 3 Users points will increase (the amount is dependant on the barcode)				
TC_O_SP_001	(TS_005) Open Sticker Pack	Verify navigating to "Store" page from the "Main Menu" screen	Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Store" from "Main Menu" screen (Verify ER - 1)	Not applicable	1 User should be navigated to "Store" page				
TC_O_SP_00	(TS_005) Open	Verify opening a sticker pack which the user has enough	1 Open the app [what the app is called] and navigate	1 Click on "Store" from "Main Menu" screen (Verify ER - 1)	Sticker Pack : "100 point sticker pack"	1 User should be navigated to "Store" page				


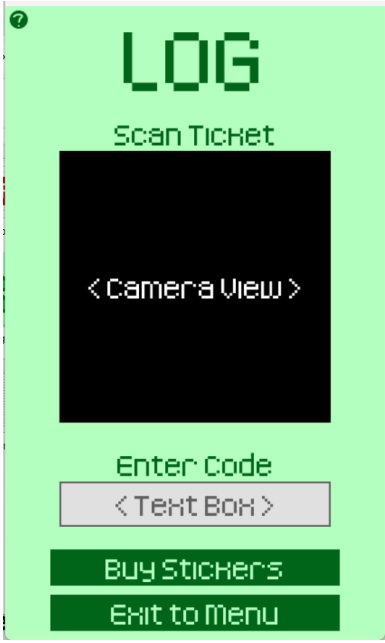
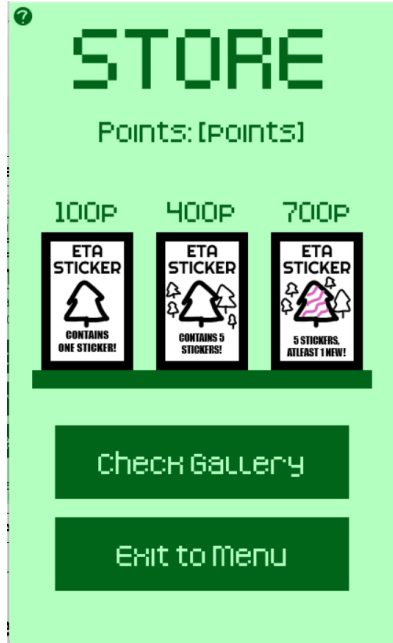
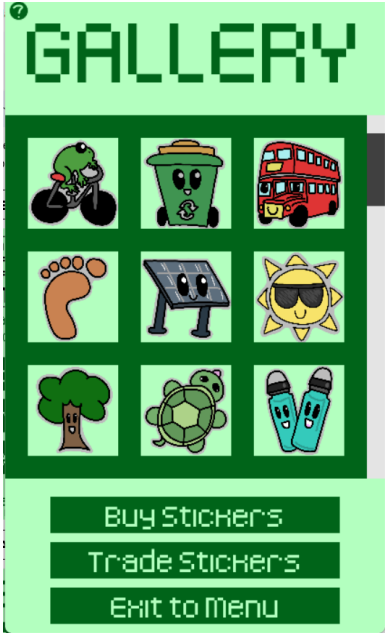

2	Sticker Pack	points to purchase	to the "Main Menu" page 2 User has 150 points	2 Click on one of the sticker packs <Refer Test Data> (Verify ER - 2)		2 User should be able to open the pack which was selected and the stickers from the pack should be shown on screen				
TC_O SP_003	(TS_005) Open Sticker Pack	Verify opening a sticker pack which the user does not have enough points to purchase	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 User has 50 points	1 Click on "Store" from "Main Menu" screen (Verify ER - 1) 2 Click on one of the sticker packs <Refer Test Data> (Verify ER - 2)	Sticker Pack : "100 point sticker pack"	1 User should be navigated to "Store" 2 Message should be displayed stating that the user does not have enough points to open the pack selected				
TC_O SP_004	(TS_005) Open Sticker Pack	Verify once the user opens a sticker pack that the points are removed from the users points	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 User has 150 points	1 Click on "Store" from "Main Menu" screen (Verify ER - 1) 2 Click on one of the sticker packs <Refer Test Data> (Verify ER - 2) and (Verify ER - 3)	Sticker Pack : "100 point sticker pack"	1 User should be navigated to "Store" page 2 User should be able to open the pack which was selected and the stickers from the pack should be shown on screen 3 Users points should decrease to 50 points				
TC_O SP_005	(TS_005) Open Sticker Pack	Verify that when a user receives a new sticker out of a sticker pack it is added to the users stickers	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2. User does not already own sticker from pack 3 User has 150 points	1 Click on "Store" from "Main Menu" screen (Verify ER - 1) 2 Click on one of the sticker packs <Refer Test Data> 3 Click on 'yes' to open the pack (Verify ER - 2), (Verify ER - 3) and (Verify ER - 4)	Sticker Pack : "100 point sticker pack"	1 User should be navigated to "Store" page 2 User should be able to open the pack which was selected and the stickers from the pack should be shown on screen 3 Users points should decrease to 50 points 4 Users stickers should be updated				
TC_O SP_006	(TS_005) Open Sticker Pack	Verify that when a user receives a duplicate sticker out of a sticker pack it is added to the users stickers	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2. User already owns sticker from pack 3 User has 150 points	1 Click on "Store" from "Main Menu" screen (Verify ER - 1) 2 Click on one of the sticker packs <Refer Test Data> 3. Click on 'yes' to open the pack (Verify ER - 2), (Verify ER - 3) and (Verify ER - 4)	Sticker Pack : "100 point sticker pack"	1 User should be navigated to "Store" page 2 User should be able to open the pack which was selected and the stickers from the pack should be shown on screen 3 Users points should decrease to 50 points 4 Users stickers should be updated				
TC_VS_001	(TS_006) View Stickers	Verify navigating to "Gallery" page from the "Main Menu" page	1 Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Gallery" from "Main Menu" screen (Verify ER - 1)	Not applicable	1 User should be navigated to "Gallery"				
TC_VS_002	(TS_006) View Stickers	Verify that stickers on the "View Stickers" page are ordered in the correct format	1 Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Gallery" from "Main Menu" screen (Verify ER - 2)	Not applicable	1 User should be navigated to "Gallery" 2 Stickers should be displayed in the correct format				
TC_TS_001	(TS_007) Trade Stickers	Verify navigating to "Trade Stickers" page from the "Main Menu" page	1 Open the app [what the app is called] and navigate to the "Main Menu" page	1 Click on "Gallery" from "Main Menu" screen 2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1)	Not applicable	1 User should be navigated to "Trade Sticker"				
TC_TS_002	(TS_007) Trade Stickers	Verify entering a username which is valid	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 Username is already saved as a valid username	1 Click on "Gallery" from "Main Menu" screen 2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1) 2 Click on text box and enter valid username <Refer Test Data> (Verify ER - 2)	Username : EpicGamer63	1 User should be navigated to "Trade Sticker" 2 User should be allowed to send sticker to other user				
TC_TS_003	(TS_007) Trade Stickers	Verify entering a username which is not valid (as it does not exist)	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 Username is not saved as a username and therefore is an invalid username	1 Click on "Gallery" from "Main Menu" screen 2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1) 2 Click on text box and enter an invalid username <Refer Test Data> (Verify ER - 2)	Username : EpicGamer100	1 User should be navigated to "Trade Sticker" 2 User should be not be allowed to send sticker to other user				
TC_TS_004	(TS_007) Trade	Verify once a user has sent a sticker to another user it is	1 Open the app [what the app is called] and navigate	1 Click on "Gallery" from "Main Menu" screen	Username : EpicGamer63	1 User should be navigated to "Trade Sticker"				

	Stickers	removed from their stickers	to the "Main Menu" page 2 Username is already saved as a valid username 3 User must already have stickers to trade	2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1) 3 Click on text box and enter valid username <Refer Test Data> (Verify ER - 2) 4 User should be click on sticker/s and press "Complete Trade" (Verify ER - 3)		2 User should be allowed to send sticker to other user 3 Users stickers which have been sent should be removed from users stickers				
TC_PP_001	(TS_008) Processing Power	Verify that the app only uses 20% of the devices processing power whilst running in the background	1 Have the app [what the app is called] open in the background	Not applicable	Not applicable	1 App should use 20% of devices processing power				
TC_PP_002	(TS_008) Processing Power	Verify that the app only uses 20% of the devices processing power whilst itself and 5 other apps are running the background	1 Have the app [what the app is called] open in the background 2 Have 5 other apps open in the background <Refer Test Data>	Not applicable	5 Apps running in background : Gmail Snapchat PokemonGo Canvas Instagram	1 App should use 20% of devices processing power				
TC_M_M_001	(TS_009) Memory Management	Verify that the app only uses 10% of the devices memory whilst running in the background	1 Have the app [what the app is called] open in the background	Not applicable	Not applicable	1 App should use 10% of devices memory				
TC_M_M_002	(TS_009) Memory Management	Verify that the app only uses 10% of the devices memory whilst itself and 5 other apps are running in the background	1 Have the app [what the app is called] open in the background 2 Have 5 other apps open in the background <Refer Test Data>	Not applicable	5 Apps running in background : Gmail Snapchat PokemonGo Canvas Instagram	1 App should use 10% of devices memory				
TC_O_SI_001	(TS_010) Operating Systems Interacting	Verify that a user can interact with another user who uses a device with the same OS as themselves	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 One user must be logged into the app using an android 3 The other user must be logged into the app using an android 4 Username is already saved as a valid username 5 User must already have stickers to trade	1 Click on "Gallery" from "Main Menu" screen 2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1) 3 Click on text box and enter valid username <Refer Test Data> (Verify ER - 2) 4 User should be click on sticker/s and press "Complete Trade" (Verify ER - 3)	User 1 is using an android device User 2 is using an android device Username : EpicGamer63	1 User should be navigated to "Trade Sticker" 2 User should be allowed to send sticker to other user 3 Users stickers which have been sent should be removed from users stickers				
TC_O_SI_001	(TS_010) Operating Systems Interacting	Verify that a user can interact with another user who uses a device with a different OS as themselves	1 Open the app [what the app is called] and navigate to the "Main Menu" page 2 One user must be logged into the app using an android 3 The other user must be logged into the app using an apple device 4 Username is already saved as a valid username 5 User must already have stickers to trade	1 Click on "Gallery" from "Main Menu" screen 2 Click on "Trade Stickers" from "Gallery" screen (Verify ER - 1) 3 Click on text box and enter valid username <Refer Test Data> (Verify ER - 2) 4 User should be click on sticker/s and press "Complete Trade" (Verify ER - 3)	User 1 is using an android device User 2 is using an apple device Username : EpicGamer63	1 User should be navigated to "Trade Sticker" 2 User should be allowed to send sticker to other user 3 Users stickers which have been sent should be removed from users stickers				

E1: Prototype

The prototype can be found [here](#). It may not work on older versions of Safari.

In case it cannot be accessed or the prototype doesn't work, screenshots of the prototype can be found in the table below, with the details given on the original page found beneath.

MAIN MENU	LOG ACTIVITY	STICKER STORE
		
STICKER GALLERY	TRADE STICKERS	
		

Main Menu

This is the first screen the user will see after logging in. In a completed project, the user's username would be displayed where "[username]" is, and the user's points would be displayed where "[points]" is.

From here in the prototype, the user can access any of the Log Activity screen (from the "Log Activity" button), the Sticker Store screen (from the "Store" button), or the Sticker Gallery screen (from the "Gallery" button). In a completed version of the app, the user would also be able to open the "Main Menu Help Page" by tapping on the "?" icon in the top-left corner (this help page would contain information on what each of the other pages do).

Log Activity

This is the screen used for the user to enter their public transport tickets, either by scanning the QR code barcode on the ticket or by entering the ticket's code. In a completed project, the view from the system's camera would appear where "<Camera View>" is, and a textbox would appear where "<Text Box>" is.

From here in the prototype, the user can access either the Sticker Store screen (from the "Buy Stickers" button) or the Main Menu screen (from the "Exit to Menu" button). In a completed version of the app, the user would also be able to open the "Log Activity Help Page" by tapping on the "?" icon in the top-left corner (this help page would explain how to use the QR/Barcode scanner, and what code should be entered into the box below).

Sticker Store

This is the screen used for the user to exchange their points for stickers, choosing from one of the three options (1 sticker, 5 stickers, or 5 stickers with at least one brand new). In a completed project, the user's points would be displayed where "[points]" is.

From here in the prototype, the user can access either the Sticker Gallery screen (from the "Check Gallery" button), or the Main Menu screen (from the "Exit to Menu" button). In a completed version of the app, the user would also be able to open the "Sticker Store Help Page" by tapping on the "?" icon in the top-left corner (this help page would explain what each of the three sticker packs contain).

Sticker Gallery

This is the screen used to allow the user to view all the stickers they have collected. In a completed project, the scroll bar on the right would allow the user to scroll through their gallery (in the prototype only a 3x3 square of stickers is shown, to represent what the gallery would look like).

From here in the prototype, the user can access any of the Sticker Store screen (from the "Buy Stickers" button), the Trade Stickers screen (from the "Trade Stickers" button), or the Main Menu screen (from the "Exit to Menu" button). In a completed version of the app, the user would also

be able to open the “Gallery Help Page” by tapping on the “?” icon in the top-left corner (this help page would explain how to scroll through the gallery, and how to get more stickers).

Trade Stickers

This is the screen that allows the user to trade their stickers with another user. In a completed project, a text box would appear where “<Text Box>” is. Like the gallery, the scroll bar on the right would allow the user to scroll through the shown stickers, and they would be able to select one by tapping on them (in the prototype only a 3x2 box of stickers is shown to represent what it would look like, and they cannot be selected). Lastly, in a completed project the user would be able to send the collected sticker to another user by pressing the “Complete Trade” button.

From here in the prototype, the user can access the Sticker Gallery page (from the “Exit to Gallery” button). In a completed version of the app, the user would be able to open the “Trade Help Page” by tapping on the “?” icon in the top-left corner (this help page would explain how trading works).

E2: Video

The video has been submitted separately as an mp4 on the assignment page on Canvas.

F: Ethics and Professional Practice

Our proposed app aligns closely with the principles of the IEEE/ACM Software Engineering Code of Ethics.

Public Interest

To act consistently with public interest, the proposed app's main purpose is to advocate sustainable practices, such as eco-friendly transportation and physical activities. The app positively contributes to the community by motivating users to use eco-friendly transportation and engage in physical activities, aligning with public interest.

Client and Employer

Our app prioritises user interests (as the client) by ensuring that it respects the user's privacy, creating a secure experience, this is done by adhering to privacy regulations, clearly communicating terms and conditions, and keeping users' data secure. By ensuring this, our app respects the best interests of the user and also contributes to increased user engagement, which in turn enhances the app's reputation, serving the best interests of employers as well.

Product

Our app is designed with high professional standards, such as the transparent communication of terms and conditions, as well as strictly adhering to data protection acts, ensuring that these standards are met. Furthermore, the implementation of robust data encryption keeps user's data secure especially when the user connects their account to social media, our app does not share any sensitive information to it without the user's permission. This shows a high level of commitment to providing a quality product while respecting ethical standards.

Judgement

Our app maintains integrity and independence throughout, with many ethical considerations, such as the impact on the environment, and the decision to allocate points for sustainable activities was made independently and considers the positive impact it would have on users and the environment.

Management

Project management practices are implemented in our app, providing an ethical approach, by integrating sustainability goals into the development process. Furthermore, throughout the development process, ethical reviews will be conducted to check how the app impacts environmental and social well-being.

Profession

Our app showcases the positive impact of technology on sustainable living, and thus contributes to advancing the integrity and reputation of the software engineering profession, displaying it positively for society.

Colleagues

Fairness and support in collaboration are important, and through our app's development, we have achieved this, by placing importance on fair decision-making, and ensuring that a supportive environment is carried throughout the development process.

Self

Software engineers will continue to be engaged in our app's development, pursuing lifelong learning, as ongoing education on new technologies and ethical considerations will be prioritised, ensuring that our app remains a leading example of sustainability and ethical design.

In conclusion, our sustainable app goes above and beyond to not only align with the principles of the IEEE/ACM software engineering code of ethics but that it is a leading example of ethical design, showing how apps and technology can be used for a positive environmental and societal impact.

Product Management & Moderation

Name	ID Number	Email Address
Rory Simpson	2454453	rxs1099@student.bham.ac.uk
Daniel Lawson	2428894	dxl295@student.bham.ac.uk
Caleb George	2321112	ceg212@student.bham.ac.uk
Frederick Foster	2429087	fjf287@student.bham.ac.uk
Conor Galvin	2431008	cxg112@student.bham.ac.uk
Daniel Jones	2456299	dxj299@student.bham.ac.uk
Allen Jen Joseph	2549623	ajj223@student.bham.ac.uk
Lilliana Etheridge	2433639	lxe239@student.bham.ac.uk

The team managed to complete the project within the deadline by working together to split tasks amongst us, and then each task was checked by other members of the group upon completion to ensure consistency between diagrams. We also took advantage of the feedback sessions, having a total of 5 sessions across the 7 weeks of the project. In order to make sure each member of the group was putting in the effort required, we met on campus for 2 hours each week where we discussed the project, worked together on sections, and divided the tasks to be done before the following week's meeting. Below is a list of contributions made by the team members:

- Daniel Lawson: Acted as the team leader (organising timings for meetings/feedback sessions), wrote parts of the requirements as well as the requirement priority table, worked alongside Allen to create the sequence diagrams and ensure they were consistent with the class diagram, and created both the prototype and video.
- Daniel Jones: Worked on requirements of the system, wrote scenarios for the potential use cases of the system, in B8 created a state machine diagram for the opening of a pack, and added some tests to the test plan.
- Lilliana Etheridge: Wrote some system requirements, collaborated on the creation of the class analysis and diagrams in B5, helped in the design of the prototype and contributed to filming the video (E1 and E2), general overseeing of the document to ensure consistency in formatting/grammar.
- Frederick Foster: Made the object diagram in section B6, created the deployment diagrams and their explanations in section C, and wrote the final design choice for C3.
- Allen Jen Joseph: Wrote parts of A1 and later made it more concise, wrote the B2/B3 use case for "Scan Barcode", assisted Rory with the diagrams in B5, created the original

versions for the sequence diagrams (B7), and other proofreading to ensure the project was accurate and precise.

- Caleb George: Worked on the introduction, including research on similar systems, and wrote some of the system requirements. Also worked on two of the use cases in B2/B3, created the state machine diagram for scanning a barcode, explained features on the test plan, and wrote the code of ethics appraisal in section F.
- Rory Simpson: Worked on the requirements and specification in section A, made the noun/verb analysis table, CRC cards and both class diagrams in section B, as well as deciding on an architecture and creating both component diagrams and their explanations in section C. Ensured diagram consistency in sections B and C.
- Conor Galvin: Wrote some requirements and assumptions in section A, created the use case diagram in B1 as well as contributing towards one of the use cases in B2/B3. Designed and created both activity diagrams in B4, and heavily contributed towards the testing plan.