“Vimo.lt”

Cross-Platform Mobile Application for Gaming Community

Final report

Name: **Vidmantas Valskis**

Student ID: **20099690**

SETU

Higher Diploma in Science in Computer Science

Declaration of Authenticity

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of ‘single quotation marks’, for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (author, date) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student:
Vidmantas Valskis

Date:
3 April 2024

Acknowledgements

I would like to thank my partner, family & friends for all their support! It was a journey in which I'm really glad I had them by my side at all times.

Big thanks to all the professors in SETU that I've had pleasure of meeting and learning from – I really enjoyed it!

Titles

Commercial Title:
Vimo.lt

Academic Title:
Cross-Platform Mobile Application for Gaming Community

Preface

GitHub project repository with Flutter & back-end server submodules:
https://github.com/Bullzai/vimo-setu

Deployed back-end server:
https://flutter.vimo.lt/api

Project show entry website:
https://vimo.lt/setu

Abstract

For the past 12 years I've been hosting and developing unique game servers, mainly focused on competitive first-person shooter game called Counter-Strike. In doing so I created a community of roughly 7,000 people.

The community has grown, my services evolved, becoming a hub & a platform of sorts for like-minded people oriented around gaming. This project aims to further enhance the user experience by introducing a cross-platform mobile application which would encapsulate many other platforms created along the way, bringing them all to one place. One app, to rule them all.

Table of Contents

Table of figures

1. Introduction

1.1. Background

For the past 12 years I've been hosting and developing unique game servers, mainly focused on competitive first-person shooter game called Counter-Strike. Early on I was asked by players in-game to create a website with a forum where they could interact with each other without having to be online and that's how my gaming community was born.

The community has grown, my services evolved, becoming a hub & a platform of sorts for like-minded people oriented around gaming. This project aims to further enhance the user experience & expand its services by introducing a cross-platform mobile application that would encapsulate the services and platforms that I created along the way. Think of it like a Swiss Army knife, having all the relative things a user might need in the palm of their hands!

Additional information:

• 6,512 Total members in Forum (https://vimo.lt/forums)

• 3,740 Total members in Discord (https://vimo.lt/discord)

We became the biggest gaming community in Lithuania somewhere in 2017, but recently Valve (company that created Counter-Strike franchise) released a new version of the game built on different, Source2 engine – effectively destroying community servers like ours with no official support for them.

But we'll bounce back! Eventually...

A similar app has been developed for Mobile App Development (Native Android app) module, but that was developed with Kotlin – completely different language and framework.

I would like to keep my GitHub repo's private (there is no issues in inviting necessary people to review them) – this app will end up in Google Play Store ideally & I have plans to commercialize it with ads in the future.

1.2. Motivation

I'm a gamer myself and to no surprise, this whole community started with me. In short – I wanted proper quality servers to play in so I could have the best gaming experience on the games that I enjoyed the most.

Since I've been developing, maintaining & hosting everything for more than a decade now, it should be no surprise that I will continue in doing so as it brings joy not only for me, but for many other like-minded people.

There were countless hours invested in this project over the last 12 years, so it would only be fair if it paid for my rent as well – here is where commercialization will come in.

1.3. Objectives

Final objective is to have a working prototype of this cross-platform mobile application that would include some of the services and platforms that I developed and am still developing to this day, which may include:

- Users in-game statistics (still in development)
- Virtual currency & membership (eCommerce – ability to buy said things)
- Marketplace (still in development)
- Inventory (still in development – ability to equip items they own)

In addition to this, it should include features like News Feeds (its development began during Native Android Mobile Application module of this course), Polls & Event Calendar.

In trying to achieve these objectives, I hope to enhance the community's engagement, general experience and to expand our services even more.

I have a few other objectives that I will reach after the initial launch & submission of this application, as those are far beyond the scope of what I can do for this project initially with such a limited time. To name a few:

- Admin panel for people who are helping me to manage and maintain the whole community and platforms (forum, in-game servers, blacklists etc.).
- Reward system for administrators and users.
- Application available on Google Play & Apple Store

2. Research and Analysis

2.1. Requirements

To reach the end goal and have everything running in production, I will need to use:

Front-end	**Back-end**	**CI/CD**	**Hardware & Hosting Services**
Flutter	TypeScript/JavaScript	Jenkins	Domain
Dart	NodeJS		VPS
	MariaDB		PM2
			Nginx

App/system functional requirements:

- User authentication via Steam
- Interaction with community
  - Polls
  - Events calendar
- E-commerce
  - Virtual currency
  - Membership
- Content delivery
  - News feed
  - In-game statistics
  - Live server information
- Inventory & item management
  - Ability to equip owned items

App/system non-functional requirements:

- Security
  - TLS1.2
  - Avoid SQL injections
  - JWT Tokens
- Usability
  - Androids
  - iPhones

2.2. Front-end

I wanted to build a mobile application that majority of the community would be able to use without any hassle. Upon doing few community polls in the past, I found out that roughly more than 60% of members are using Android devices, the rest were mostly using iOS devices.

So, in order not to discriminate against the lower proportion of the community with an access to an application, which hopefully should bring them better user experience and convenience in general, I decided to create this application using Flutter framework, as I could deploy this application to multiple devices 'from a single codebase' (Flutter, 2024)

2.3. Back-end

For the last handful of years, I've been hosting my own back-end API server using NodeJS with Hapi. The back-end server serves API endpoints in JSON format for my current platforms that were developed for the community and gaming servers.

For data storage I've been using MariaDB since 2012, it only makes sense to continue using it as I like the approach and idea of relational databases. New API endpoints will need to be developed for this project to serve the necessary data for the mobile application.

Database is hosted on a bare-metal workstation that is collocated in a data center to provide better availability and security, which includes business network thruput and anti-DDoS protection as my servers were and still are under constant & heavy attacks.

Back-end API server itself is hosted on a VPS from the same data center as they offer these services as well.

2.4. CI/CD

To achieve 'continuous integration and continuous delivery' (Synopsys, 2024) I will try to use Jenkins, 'the leading open-source automation server.

Jenkins provides hundreds of plugins to support building, deploying and automating any project' (Jenkins, 2024).

Before I used to create bash scripts which would pull the code from main branch in GitHub repo, build the application, change the port & re-launch the application if needed so it would be immediately available for users/customers. I might revert back to this crude method if I find myself lacking time to implement a proper CI/CD solution.

2.5. Market analysis

This is orientated towards a specific group of people – Lithuanians who enjoy First-person Shooter games. There's a handful of other, similar projects & communities like mine:

- https://bachuruservas.lt
- https://slimi.lt
- https://rampage.lt

The difference in what we offer to them is a unique gaming experience in our hosted servers, unique systems like owning virtual currency, ability to use said currency for buying virtual items* which they can equip* or sell* again and the most recent addition – ability to open cases and win items, just like in any other major game, for example: Counter-Strike 2, Apex Legends. The proper term for this would be loot boxes.

* - Still in development & hopefully will finish development during this project.

After searching for similar idea-based mobile applications on Google Play & GitHub, I learned that there were no such applications out there. I specifically looked into direct competitors and few other similar communities – none of them have a mobile application of such sort, which makes me think it will indeed enhance user's experience by offering something new and unique to them, that they cannot find anywhere else.

Regarding technical side of application – there are many applications that would have similar functionality to what I have in mind, which may include:

- Calling API endpoints and displaying data upon receiving response.
- Signing in via Steam OpenID.
- Poll voting
- Sending data to API endpoints
- Navigation (drawer)
- Gesture detection

3. Modeling

3.1. Back-end

For back-end server I will be using these technologies:

- Node.js
- JavaScript
- TypeScript
- Hapi.js

Server itself will be hosted on a VPS.

It will communicate with relational MariaDB database using Knex.js SQL query builder (Knex, 2024) and provide required data for the mobile application using JSON format.

Some data, like server statistics and current player lists of game servers will be live data, but only if I will have enough downtime to fix server queries using S2C_CHALLENGE (Valve Developer Community, 2024), which according to their documentation - they should.

Though documentation itself was not updated after the release of Counter-Strike 2, which might explain why their own official browser fails to provide such information upon querying servers, despite it being a standard for the last two decades.

I already had an existing back-end server, which was developed with python, but upon doing Full Stack Web Development assignment #2 I really liked the approach and design that was introduced to me by this course and I decided to adopt it.

After the submission of the assignment, I continued to develop the back-end server: implemented my old API endpoints, created new ones, developed new utilities and new features/logic for other platforms that were developed. That back-end server is still in development and quite messy even though it's being used in production; I would not feel comfortable sharing the code/application with anyone as it would require quite a lot of attention to tidy it up & in general it has a lot of code that would not be relative to this project. For that reason, I will create a separate, new back-end server for the Flutter application as it would be more practical, tidy, less shameful and a lot more easier for everyone to read.

3.2. Database

The only data that will be collected by the mobile application will be the results of the polls.

The application will mostly be showing the data that I collected from my gaming servers over the past 12 years.

Most of the tables were not created & designed by me, they were created by applications/software's that were used over that period of time, for example - the forum that is being used at my website https://vimo.lt is using forum software from Invision Community which created 207 tables as depicted in the figure 1 below:

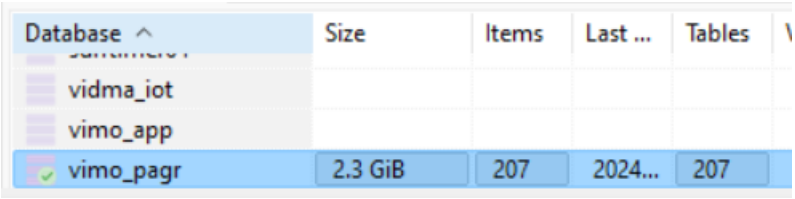

Figure 1 - Database table created by software

So far, I created a database with utf8mb4_unicode_ci collation so I could have special characters stored in the database, which are used within Lithuanian language. The database is based on InnoDB engine and I've created a table for news feeds - it's ER diagram you can see below on figure 2:

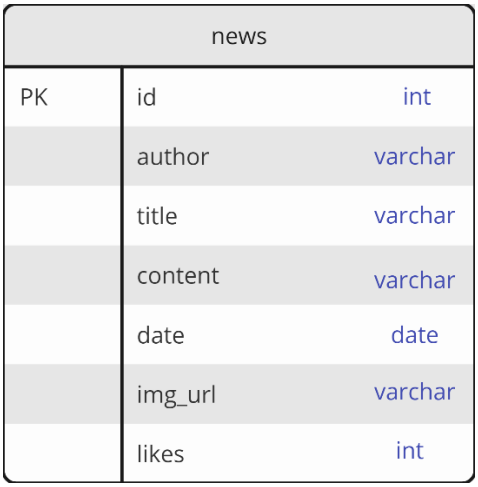

Figure 2 - Database table's ER diagram

Once I reach poll feature in the implementation phase, I will create a table to keep the poll data as well.

3.3. System architecture

Design of the system is relatively straight forward – user can use the mobile application regardless if he's logged in or not. Logging in will provide more features and better user experience for the user.

User can login only via Steam – a login request is sent to the back-end server -> user is redirected to Steam's authentication servers -> Steam sends back the user's info back to back-end server -> back-end server generates a token for the user, fetches additional data from database associated with the user and sends everything back to the mobile application -> user can see statistics & virtual currency he acquired.

To visualize it I created a system model, which you can see below on figure 3:

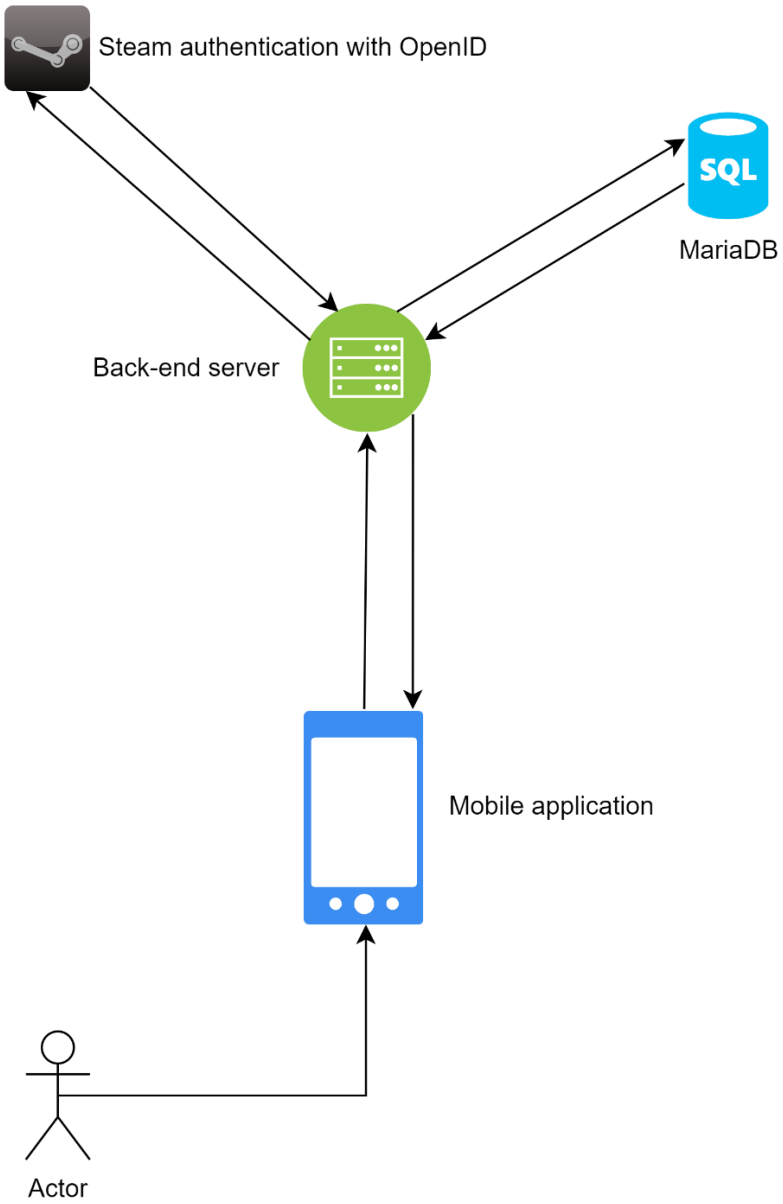

Figure 3 - System model

3.4. Front-end

So far, I only have created two API endpoints – news feed & servers list. The design will definitely change during the implementation and development phase, but the rough idea & looks can be observed on the wireframe below that I drew using Figma.com:

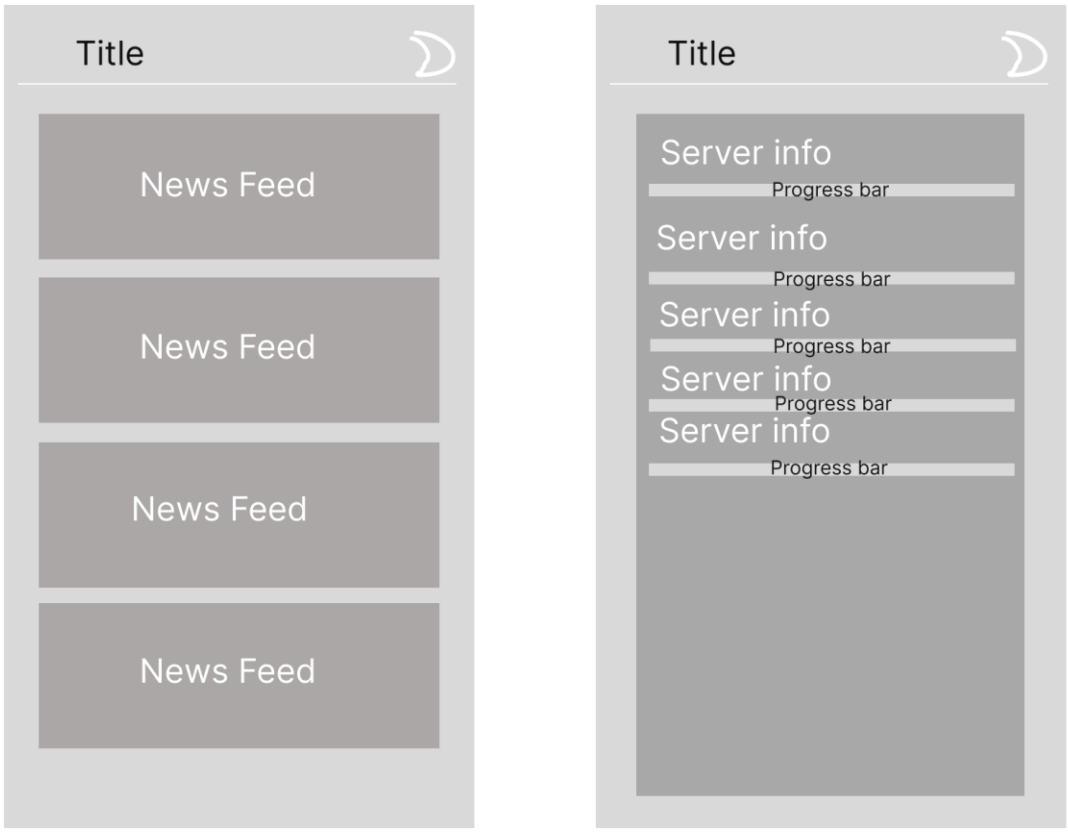

Figure 4 - News feed & servers list wireframe

The moon shape icon at the top right corner represents a button, which will change the theme from light to dark. Moon icon to change to dark theme & sun icon to change to light theme.

4. Planning

At my current workplace we're using Trello with Kanban approach to organize work that needs to be done on any type of project that is thrown our way. We're doing them in 2-week sprints, which aligns to the guidelines for this project as well.

It only makes sense to use the same approach here as well since I like the logic behind it and it actually helps your teammates to see what you're currently working on and what has already been completed.

In short, I'll be using Agile methodology with Kanban and 6 sprints of 2 weeks.

Current Trello board is constantly being adjusted, so far you can have a sneak peek on figure 4 below (moved Sprint5 and Sprint6 over to the other side so have better visibility for figure 4):

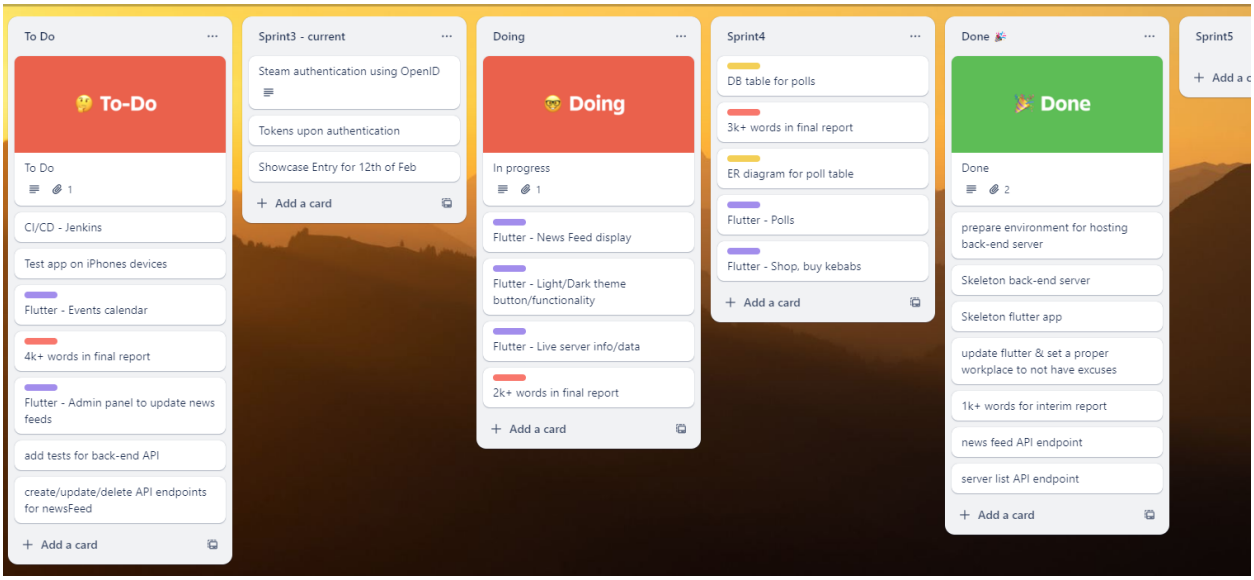

Figure 5 - Trello board with sprints

5. Implementation

5.1. Sprint #1 (Jan 7 – Jan 20)

During first sprint I mainly focused on RAMP – research, analysis, modelling and planning.

5.2. Sprint #2 (Jan 21 – Feb 3)

In this sprint I created a skeleton back-end server using NodeJS, TypeScript & Hapi framework.

MariaDB database is already in use for data persistence in my infrastructure for the last 9 years, naturally I chose to use it for this project as well. Knex & Dotenv packages were used in the back-end server to establish connection with the database & the first API endpoint was created – News Feeds.

Upon successfully returning the feed data on News Feeds endpoint I created a skeleton Flutter application with one view – News Feeds. It retrieves the feed data from the back-end server and displays it in a nice manner using FutureBuilder & Card widgets.

5.3. Sprint #3 (Feb 4 – Feb 17)

During the beginning of sprint #3 I mostly focused on writing the Interim Report.

Right after submitting Interim Report I started working on one of my platforms, Marketplace (https://market.vimo.lt) – I needed to release an update with new functionality for the community as they were asking for it for quite some time now – I also intended for some of the functionality to be available on this application as well, and having it tested by the community would be a great before I go ahead and implement it here as well, as I would like my first mobile application release (on Google Play / Apple Store) to have as little bugs as possible so it wouldn't leave bad first expressions among other things, of course.

Functionality I was working on consisted of:

- Post item.
- Cancel posted item.
- Buy posted item.
- Equip owned item.

Some of the functionality that I developed for Marketplace during this sprint I will implement in the Flutter application on Sprint 4, 5 & 6.

During the release of this update, I've managed to get myself very important and relevant experience that I will put to use for further development of this project.

Not 30 minutes after the initial release, I realized that I've missed quite a few probable use cases that users might encounter. After receiving few complaints and reports, I've started monitoring error logs.

While I was patching my code for the back-end server to reflect on those reports and fix the unforeseen use case scenarios, I noticed a specific error that made my heart skip a beat – I thought I finally fell victim to SQL Injection, see Figure 6 below:

```
TypeError: Cannot destructure property 'id' of 'request.payload.item' as it is null.
    at handler (file:///home/vimo/web/market.vimo.lt/market-hapi/src/api/market-api.ts:329:17)
    at exports.Manager.execute (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/@hapi/hapi/lib/toolkit.js:57:29)
    at Object.internals.handler (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/@hapi/hapi/lib/handler.js:46:48)
    at exports.execute (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/@hapi/hapi/lib/handler.js:31:36)
    at Request._lifecycle (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/@hapi/hapi/lib/request.js:370:68)
    at processTicksAndRejections (node:internal/process/task_queues:95:5)
    at Request._execute (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/@hapi/hapi/lib/request.js:280:9)
Error: update `projecttm_wsgk_ws` set ``test` = '' where `steamid` = 'STEAM_1:1:640600554' - Unknown column ''test' in '
field list'
    at Packet.asError (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/packets/packet.js:728:17)
    at Query.execute (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/commands/command.js:29:26)
    at Connection.handlePacket (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/connection.js:489:32)
    at PacketParser.onPacket (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/connection.js:94:12)
    at PacketParser.executeStart (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/packet_parser.js:75:
16)
    at Socket.<anonymous> (/home/vimo/web/market.vimo.lt/market-hapi/node_modules/mysql2/lib/connection.js:101:25)
    at Socket.emit (node:events:514:28)
    at addChunk (node:internal/streams/readable:376:12)
    at readableAddChunk (node:internal/streams/readable:349:9)
    at Socket.Readable.push (node:internal/streams/readable:286:10) {
  code: 'ER_BAD_FIELD_ERROR',
  errno: 1054,
  sqlState: '42S22',
  sqlMessage: "Unknown column ''test' in 'field list'",
  sql: "update `projecttm_wsgk_ws` set ``test` = '' where `steamid` = 'STEAM_1:1:640600554'"
}
```

Figure 6 – SQL injection attempt

As you can see from the Figure 6, my back-end server reported that it tried to execute SQL query which was invalid. When I took a better look at the content of these errors, I realized a user was manipulating

POST requests generated by the front-end Marketplace application (created with SvelteKit) - he changed the data inside the request and sent it through to my back-end server.

In doing so he managed to do... some interesting things & get access he was not supposed to have.

Few examples of what he did:

- Equipped an item he did not own.
    - Equipped an item for another user that he or the user did not own.
- Bypass all limitations I've put in the front-end Svelte application.
    - Effectively causing a lot of errors being thrown by the back-end server.
    - SQL queries error out, as he was exceeding limitations.
- Posted an item pretending to be someone else.
    - Posten an item with negative price value on it.
- Posted an item he did not own.

Some of the things he did was quite funny, some of it was quite nasty. With the help of the community, I did manage to get in touch with this user and he turned out to be not that evil at all!

He admitted in using Burp Suite and intercepting the requests just as I've suspected - according to the user it's his hobby to play around with web exploitation, he meant no harm with it as he was planning to report it all, once he was done – so he said!

While I was implementing additional checks, requirements, validation & refactoring some of my logical flows I've asked him to continue doing it and find as many vulnerabilities as he could on my platform and report it back to me.

The update was released midday on Friday, patching of the code was finished Sunday in the evening. Very interesting weekend, it was!

Knowledge & experience I've gained from pushing the update:

- No matter which front-end application sends data to your back-end server, validate it!
    - Validate the type; the content; even the access for the data.
- Make sure to parameterize your SQL queries to avoid SQL injection.
    - Use libraries that will do that automatically, like Knex.
        - Make sure to read about them, because some functions they offer are still vulnerable to SQL injections.
    - Never use the received data directly in the queries, always validate it even if they are parameterized.
    - Use as little of the received data as possible when building your SQL queries.
- Double check your logical flow of functions and checks.
- Having proper logging and error handling will help you resolve issues faster.
- Having tests for all possible scenarios for your API endpoints would definitely help to prevent this, will have to work extra hard on this!

5.4. Sprint #4 (Feb 18 – Mar 2)

Flutter updates during this sprint:

- User authentication via Steam's OpenID with the help of webview_flutter package (flutter.dev, 2024).
- Refactored drawer widget utility from StatelessWidget to StatefulWidget to reflect changes upon logging in and logging out.
    - User information is shown in the drawer header.
- Implemented user's theme selection persistence.
- JWT token (Auth0, 2024) successfully passed and stored with the users' info in SharedPreferences (Flutter, 2024).
- Introduced a new state management package – Provider (Rousselet, 2024).
    - Refactored Drawer utility to use this new package & reflect changes in DrawerHeader upon logging in & logging out.
- Created a new view – Shop, users can:
    - Select a package for virtual currency that they'd like to buy.

In Flutter application I used skins.json (Islam, 2024) from https://github.com/Nereziel/cs2-WeaponPaints/blob/main/website/data/skins.json to get items data, like their id's & image URLs (I've based my Marketplace platform over this same skins.json data).

- Inventory - new view for user to see the items they own in gaming servers.
    - Only logged in users can see & access this information.
    - ShowDialog (Flutter, 2024) was used to display data upon tapping on an item.
    - Users have an option to equip the item.
- Created API calls utility to hold all of the endpoint calls in one place.
- Refactored Inventory UI & UX and implemented search functionality as it's a must for users with more than 10+ items.
    - At first tried to do it without any packages, ended up using AnimatedSearchBar (lazycatlabs.com, 2024) package from Flutter for its simplicity.

Back-end updates during this sprint:

Had to continuously update & deploy the back-end server for a period of time to my VPS which has nginx set-up with valid SSL certificates – WebView package in Flutter does not allow unencrypted connections and self-signed certificates. To proceed with the development of authentication via Steam I had to keep back-end server up & running in my production environment, but still in development mode.

Created new API endpoints:

- userInventory - Upon validating the token it returns users inventory data.
- equipItem - The initial logical flow was thought out during sprint #3 for the Marketplace platform, that I intended to use here. As luck may have it, a bad actor attempted to meddle with POST requests in that platform which made me refactor the code on the fly. While re-developing the endpoint for this back-end server, I refactored it again & introduced Joi data validation (Sideway, 2024).

In this sprint I finally implemented tests. At first, I tried to use Mocha and Chai, but then I started to run into quite a few issues, as my back-end server is written in TypeScript and it's configured to use ES Module syntax. After many hours of trying to implement it, I've decided to discard everything and give a shot for Lab & Code (Sideway, 2024), which is what official Hapi documentation suggests using. They are developed by the same team so the integration should be seamless. It wasn't as seamless as I'd like it to be, but it was better nonetheless.

Had to refactor the main server.ts file to accommodate test environment. Exported servers init() so it could be used with Lab & added additional check to start the actual server, so my development, production and test environments would work all together without needing to comment lines of code.

Tests are run against built application, meaning that I have to build it first before running tests.

Implemented tests:

- Public-api:
    - Tests News Feeds endpoint.
    - Tests Servers List endpoint.
- User inventory:
    - Get inventory without Token.
    - Get inventory with manipulated Token.
    - Get inventory with valid Token.
- Equip item:
    - Post request to equip item without Token.
    - Post request to equip item with Token but no item data.
    - Post request to equip item with Token & bad item data.
    - Post request to equip item without Token & good item data.
    - Post request to equip item with Token & good item data.

After running these tests I've seen that my token validation function needed an extra check to throw a proper response code & error message. It also showed me that I needed to add request & error arguments and not just h to failAction function in Joi validation part of code for it to properly return the response and stop the exceptions being thrown in back-end server console after building the server.

Quality of Life (QoL) updates:

- Fixed few bugs in Flutter application:
    - Set & cleared users steam ID upon logging in & logging out.
    - Directed user to login via Steam upon trying to purchase virtual currency when the user was not logged in.
    - Fixed a crash when user tried to log out while viewing his inventory.
- Refactored the structure of application files:
    - Instead of keeping models for specific views in the /views folder, moved all models into /models folder and the ones that use Provider with ChangeNotifier to /providers folder.

5.5. Sprint #5 (Mar 3 – Mar 16)

In sprint 5 Admin Panel was introduced and refactored, admins can:

- Add a new feed.
- Delete a feed.
- Update a feed.

Started using Provider in news feed's view.

In the back-end server had to accommodate few changes as well, like the fields returned and the actual Feed object, that helped resolving few issues in the Flutter app – the most notorious one being feed ID. Since the Flutter app doesn't send or generate the newly created feed ID, there were issues with the Feed object when the user tried editing it or deleting it. The ID is automatically generated (incremented) by database.

Changed Flutter applications name to Vimo.lt, added app icon & signed the android bundle on build with a key. Created Google Play Console developer account, had it approved & started the process of submitting this application to Google Play, see figure 7 below:

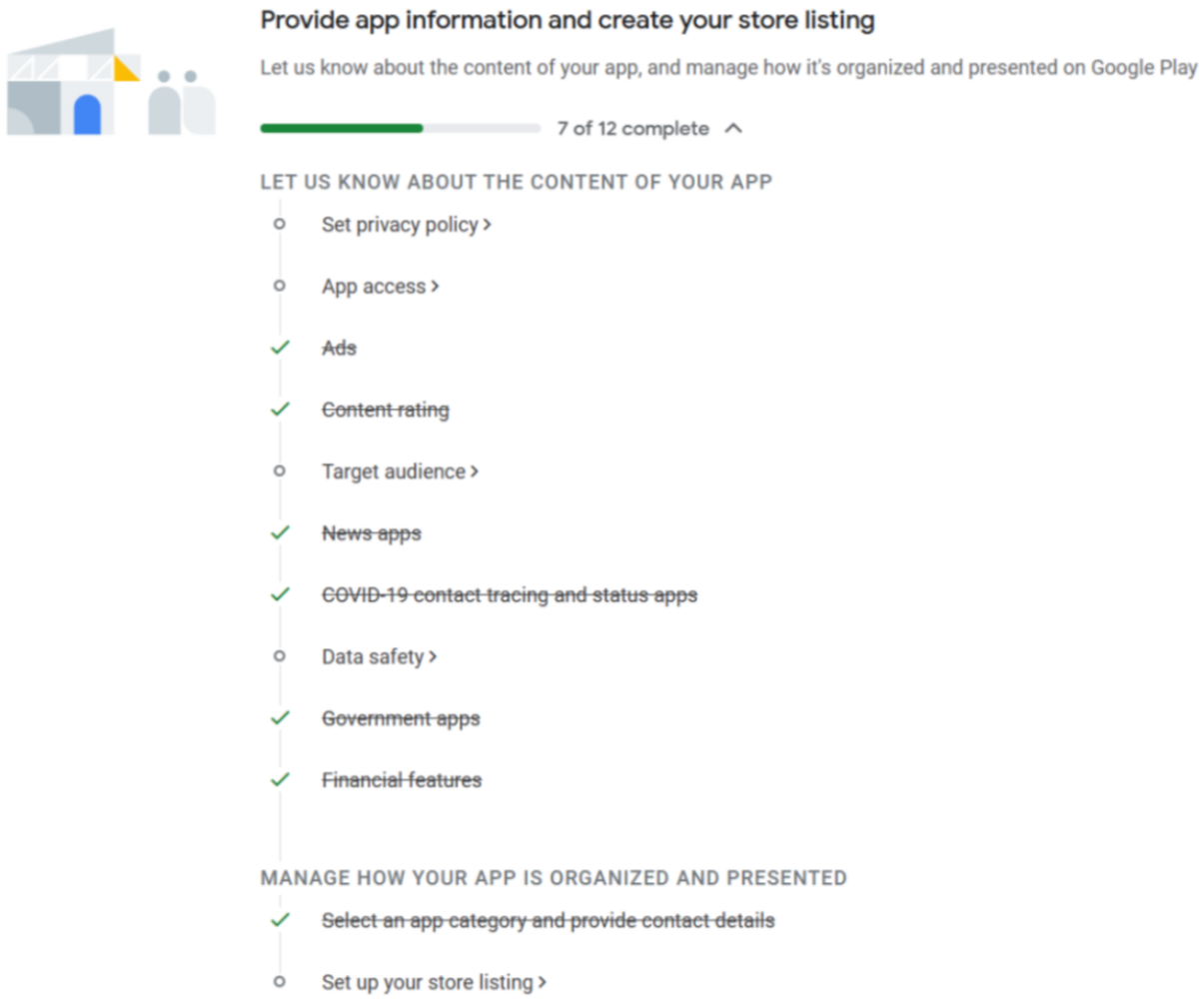

Figure 7 – Process of application submission to Google Play

Had to refactor the data structure for skins.json and agents.json files couple of times during this sprint as the in-game plugin that is used in our gaming servers drastically changed during this period, so I decided to generate my own data to be as less dependent on other platforms as possible. This data is used in Flutter applications to display item information and in the back-end server to double check and cross-reference the data validity upon requests. A GitHub repo from “ByMykel” was used to generate the custom data - https://github.com/ByMykel/CSGO-API

Added few extra options for users when they equip the item they own:

- Set Wear level of the item.
- Set Seed of the skin.
- Set a Tag on the item.
- Enable or disable StatTrack on the item.

All of the above-mentioned features are settings which modify how items look & feel in our gaming servers.

Adjusted the UI & UX of Inventory to reflect the rarity of items they own, by adding a color-coded stripe on the left side of the item.

Back-end servers’ handlers had to be fully refactored as well to reflect changes on the new JSON data structure. New database with new tables were also used in the gaming servers regarding user items, this also meant that further refactoring was needed for the handlers.

Added additional feature to application which I originally did not think of doing – Free Items.

Added ability for users to equip free items, which are of different type & cannot own currently – Agents & Gloves. This required to create new views with bottom navigation for better user experience. Back-end server additionally required a separate handler for this, as the data structures are different, different tables in database are used & different logical flow & data validation had to be done for these free items.

5.6. Sprint #6 (Mar 17 – Mar 30)

On the last sprint I implemented a new view in Flutter app – Market. In the Market, users can:

- Click on posted items:
    - Buy the item if they have enough currency in their account.
    - See a chart (Khoshabi, 2024) with previous item sales which would suggest the current going market price.
    - Search for posts or filter by price range (Flutter, 2024).
- See their own posted items in “My Posts” view which they can navigate to using bottom navigation.
    - Cancel their posted item.
- See their sold items in “My History” view to which they also can navigate using bottom navigation.

In addition to that, ability to post owned items for sale has been implemented as well in Inventory view.

For back-end server new endpoints & handlers were created to support:

- Buying items.
- Displaying chart with previous item sales data.
- Posting items.
- Cancelling items.
- History of users sold items.

Some Quality of Life (QoL) work has been done as well:

- Refactored Search in Flutter app.
    - No longer calls API endpoint each time a user uses search function.
        - Endpoint is called & data stored during widget init.
- Additional checks across the app to check if token is expired.
    - Automatically log out the user if it is.
- Refactored Inventory and Market UI & UX to provide better user experience.
- Change Theme icon moved from AppBar to Drawer as a button to improve UX.
- Added loaders (NearHuscarl, 2024) & improved success messages shown in Market view.
- Overall styling adjustments across the Flutter app.

6. Reflection

6.1. Learning

I've learned quite a number of things during this project, the most important one would be the importance of data validation & bullet-proof logic in the back-end server. Ideally, I should have spent more time on trying to think of every possible use case scenario a user might find himself in when using the application & then the same amount of time should be invested in thinking of what a bad actor could possibly do or which functions, logical flows he could use to exploit the application.

Writing tests does help, but at the same time a developer should invest time to try and penetrate his own application, using Burp Suite for example before launching it for production. It will pay off!

Along with learning on how to use Flutter & Dart programming language, State Management was another important thing. A lot of research went into FutureBuilder, Provider & other techniques on how I could manage app state & deal with user input or external data changes accordingly.

UI/UX was probably the hardest thing to learn as I'm not that much of a UX developer, but finally wrapping my head around how everything is nested inside Flutter especially when it comes to UI/UX flipped a switch in my head and I'm really glad I took upon this project because of that!

I've also learned that working in an environment where you have access to so many different packages (or libraries) can really help you to extend the functionality of application. Just make sure to choose the right packages, which would suite your needs and that are actually actively maintained!

6.2. Achievements

I'm creating this project with the intention of publishing it to Google Play, which is already in action, just need to pass the initial closed test and then 2 weeks of testing with 20 users. The objective was to have a working prototype which would enhance the community's engagement, general experience and expand my services even more – which led me to change some priorities during the development of this project regarding the features it should offer as I have to pay attention to what the community actually currently needs and prefers.

I still achieved most of the goals, some of them even went an extra mile I'd say – Not only do I have a working prototype, but an actual production ready application, at least I believe so personally. Polls & Events Calendar were not that much desired by the community after I first exposed it to the inner circle of sorts – users who either help me to manage the whole community or help me to keep it up by donating. After receiving some feedback and insights, I refocused my attention and time to developing other features accordingly.

During sprint 3 specifically I've managed to successfully achieve the community's most desired fundamental features, and not because I was pressured by them! Not at all...

It laid a good foundation for the back-end server of this project to be successfully launched for production without encountering any major issues – at least I hope so!

I was not initially planning on implementing the Admin Panel for the first public release, but eventually I realized that having a compact way to post & edit the feeds will be beneficial, as I'm planning to leave

this project alone for at least 6 months after the initial release on Google Play, to collect more feedback, make a proper plan on what to implement next (or fix) and to allow myself to work on other community related projects.

6.3. Future Development

The project will definitely be developed further after the submission. After releasing the first version of the application to production I will closely monitor user reviews and feedbacks, so I could plan accordingly what will have to be fixed or implemented next.

The gaming community is growing every day and more platforms are being introduced & used by the community - the most logical approach would be to continue integrating those platforms so users could have them all handy in one place, here, in this Flutter application.

So far besides the reward system, I would also love to implement a more detailed server list information – users should be able to tap on the server in Servers List & see the player names, time spent & their current statistics in the server.

Push notifications will be a top priority as well to implement, more info about it is in Problems Encountered section.

Highest on the list is to finish the Marketplace platform and functionality, that will have to available here as well:

- Buy & open cases (Loot boxes) to win items.
- Add Agents, Gloves, Stickers & Music kits into cases as well.
    - Users should be able to own them & trade them just like weapons & knifes currently.
- Update all images from Github links to Valve's CDN – better quality & will load a lot faster.
- Auto-update feature when Valve releases new type items or skins.

Would be nice if users could leave reactions/likes on the feeds as well – not necessary though, only nice.

6.4. Problems Encountered

The biggest problem encountered was implementation of push notifications. My current workstation on which I am developing this project is not in very good shape – the environment paths were lost and a good bunch of other things happened to it – it direly needs to be wiped & set-up again from scratch, unfortunately I did not have time or will to do that during this course.

I spent quarter of a sprint trying to implement push notifications to the Flutter application, but in the end, I gave up and decided to come back to it once my workstation situation is sorted – was having issues running Firebase CLI on my machine, which was the biggest blocker to implement this feature.

Another problem was application state management to display appropriate data for the user. After reading about many packages Flutter environment offers for these type of issues – I decided to go with Provider package. A specific example would be the User class – whenever any of the Users fields are updated, Token for example – the widgets that are "watching" User will be rebuild and reflect the changes made to the actual user using the application.

At the very end I had to invest some time into dealing with expired tokens, as I wanted users not to be flabbergasted when their Inventory or Market wouldn't load:

- Adjusted the verifyToken function in back-end server to respond with 418 error code when Token is expired.
- ApiCalls class was adjusted in Flutter to throw a custom TokenExpiredException exception on every endpoint call that used a token.
    - Views were adjusted – API calls were wrapped in a Try Catch
        - On a TokenExpiredException exception user is forcefully logged out

It's not the most graceful solution and will most likely be refactored in the future, but it's better this way & acceptable for production.

Was having issues setting up Testing environment with Mocha & Chai initially, mostly because the back-end server is written in TypeScript. After spending an evening trying to make it to work properly, I gave a shot to officially recommended Code & Lab and it worked a lot better for me.

Time limitation & management was another big problem – I could definitely work on my time management & the result of not having it under control was CI/CD. I did not leave enough time to implement proper CI/CD solutions & had to abandon it in order to reach other goals, or rather desires – having a production ready application.

7. Bibliography

Auth0, (2024). *jsonwebtoken - npm.* [Online]
Available at: https://www.npmjs.com/package/jsonwebtoken
[Accessed 23 February 2024].

flutter.dev, (2024). *webview_flutter example | Flutter package.* [Online]
Available at: https://pub.dev/packages/webview_flutter/example
[Accessed 21 February 2024].

Flutter, (2024). *Build apps for any screen.* [Online]
Available at: https://flutter.dev/
[Accessed 23 Jan 2024].

Flutter, (2024). *RangeSlider class.* [Online]
Available at: https://api.flutter.dev/flutter/material/RangeSlider-class.html
[Accessed 18 March 2024].

Flutter, (2024). *showDialog function - material library - Dart API.* [Online]
Available at: https://api.flutter.dev/flutter/material/showDialog.html
[Accessed 28 February 2024].

Flutter, (2024). *Store key-value data on disk.* [Online]
Available at: https://docs.flutter.dev/cookbook/persistence/key-value
[Accessed 20 February 2024].

Islam, S., (2024). *how to read local json file in flutter.* [Online]
Available at: https://stackoverflow.com/questions/71294190/how-to-read-local-json-file-in-flutter
[Accessed 19 February 2024].

Jenkins, (2024). *Build great things at any scale.* [Online]
Available at: https://www.jenkins.io/
[Accessed 24 Jan 2024].

Khoshabi, I., (2024). *FL Chart App.* [Online]
Available at: https://app.flchart.dev/
[Accessed 20 March 2024].

Knex, (2024). *SQL query builder.* [Online]
Available at: https://knexjs.org/
[Accessed 24 Jan 2024].

lazycatlabs.com, (2024). *AnimatedSearchBar.* [Online]
Available at: https://pub.dev/packages/animated_search_bar
[Accessed 28 February 2024].

NearHuscarl, (2024). *Flutter - How can I add a circular loading indicator to my button?.* [Online]
Available at: https://stackoverflow.com/questions/63234780/flutter-how-can-i-add-a-circular-loading-indicator-to-my-button
[Accessed 25 March 2024].

Rousselet, R., (2024). *provider - Dart API docs.* [Online]
Available at: https://pub.dev/documentation/provider/latest/
[Accessed 24 February 2024].

Sideway, (2024). *lab v25.2.0 - hapi.dev.* [Online]
Available at: https://hapi.dev/module/lab/api/?v=25.2.0#test-driven-development
[Accessed 2 March 2024].

Sideway, (2024). *Validation - hapi.dev.* [Online]
Available at: https://hapi.dev/tutorials/validation/?lang=en_US
[Accessed 1 March 2024].

Synopsys, (2024). *What Is CI/CD.* [Online]
Available at: https://www.synopsys.com/glossary/what-is-cicd.html#:~:text=Definition,are%20made%20frequently%20and%20reliably.
[Accessed 24 Jan 2024].

Valve Developer Community, (2024). *Server queries.* [Online]
Available at: https://developer.valvesoftware.com/wiki/Server_queries#Source_Server
[Accessed 24 Jan 2024].

Appendix A – Ethics Checklist

This Ethics Checklist must be completed for all final year undergraduate, taught postgraduate and research projects in the School of Science and Computing.

View your response(s)

Respondent: **Vidmantas Valskis** (Group: CM-HDIPCS) Submitted on: Saturday, 18 November 2023, 3:18 PM

Ethics Checklist for Undergraduate, Taught Postgraduate and Research Projects in the School of Science and Computing

All students in the School of Science and Computing who are either (1) in the final year of an undergraduate/BSc degree, or (2) on a taught postgraduate/MSc programme **must complete this Ethics Checklist before conducting their project** regardless of the project type or discipline. The Checklist should also be completed by anyone (whether staff member or student) conducting a **research project** (whether programmatic or not) within the School.

The purpose of this Ethics Checklist is to **identify projects that will require formal ethical approval** from the School Research Ethics Committee, or the SETU Research Ethics Committee, before they can proceed.

Students/applicants should note that this Ethics Checklist is a **formal declaration**, and great care must be taken to **answer all questions accurately**. Students should consult with their project supervisors/advisors regarding any aspects or questions that they are unsure of before completing and submitting the Ethics Checklist.

Students/applicants must **answer all questions** presented to them until the Checklist questionnaire is completed.

Feedback Report

?

No human experimentation issues (UG).

No animal experimentation issues (N/A).

No issues regarding the use of human tissues.

No animal tissue or biological fluids issues.

No ionising radiation issues.

No primary data collection issues (N/A).

No underage/vulnerable people issues (UG).

No issues regarding existing/secondary data use (public anonymous and/or with explicit consent).

Note: Full compliance with the EU General Data Protection Regulation (GDPR) and the Data Protection Act (2018) is necessary for any personal data processing.

No controversial data issues.

No issues related to the collection of rare or protected plants.

No issues regarding the use of genetically modified (GM) plant material.

Instructions:

1. If the above feedback is **entirely green** then, based on your answers, there is **no need to apply for ethical approval** for your project.
2. If **any** part of the above feedback is **yellow/amber**, then there is at least one issue with your project that needs to be reviewed and **you must apply for ethical approval** to continue your project.
3. If **any** part of the above feedback is **red** then there is a serious ethical issue and **you cannot continue your project** as currently planned.

It is recommended that you print this Feedback Report to a PDF file for your records. You should also forward and discuss this Feedback Report PDF with your project supervisor. They will be able to advise if you have any further questions or if you need to apply for ethical approval.

1 * Are you a student on a **final year undergraduate** programme, a **taught postgraduate** programme, or are you conducting a **research project**?

- (•) Final Year Undergraduate
- ( ) Taught Postgraduate
- ( ) Postgraduate Research Project
- ( ) Other Research Project

2 * What is the **working title** of your project?

Vimo.It – Cross-Platform Mobile Application for Gaming Community

3 * Who are the project **supervisors/advisors/principal investigators**?

Colm Dunphy / TBC

4 * Does your project involve **human experimentation**?

- ( ) Yes
- (•) No

5 * Does your project involve **live animal experimentation**?

- ( ) Yes
- (•) No

(6) * Is the planned animal experimentation limited to **non-invasive procedures only** (such as feeding, weighing, or taking naturally voided faecal or hair samples), and does **not** involve any invasive procedures (such as taking rectal faecal samples or blood) from live animals?

- ( ) Yes
- ( ) No

7 * Does your project involve the use of **human** remains/cadavers/tissues/cells/biological fluids/embryos/foetuses?

- ( ) Yes
- (•) No

(8) * Do you intend to only use established **commercial human cell lines**, and no other **human** remains/cadavers/tissues/cells/biological fluids/embryos/foetuses in your project?

○ Yes ○ No

9 * Does your project involve the use of **animal cells, tissues** or **biological fluids**?

○ Yes ◉ No

(10) * Do you intend to only use (1) **established commercial animal cell lines**, or (2) **slaughterhouse-derived tissues/fluids**, or (3) **fluids collected as part of routine animal husbandry** (e.g. milk) and no other animal tissues or biological fluids in your project?

○ Yes ○ No

11 * Does your project involve the **collection of rare or protected plants**?

○ Yes ◉ No

12 * Does your project involve the generation or use of **genetically modified (GM) plant material**?

○ Yes ◉ No

(13) * Do you agree to (1) only use **established genetically modified (GM) plant cell lines, seeds, or plant products** in your project, (2) **not generate new plant mutations** using chemical or other means, and (3) follow specified SETU **containment and use protocols** for GM plant materials at all times?

○ Yes ○ No

14 * Does your project involve the use of **ionising radiation**? (e.g. use of gamma ray spectrometry)

○ Yes ◉ No

(15) * Do you agree to carefully **follow the instructions** of the SETU designated **Radiation Protection Officer (RPO)**, and **adhere to all legal requirements** as set out in the Radiological Protection Act 1991 (Ionising Radiation) Regulations (2019), regarding the use of ionising radiation materials and equipment?

○ Yes ○ No

16 * Does your project involve the **collection of any new (or primary) data** from **individual people or groups**?

○ Yes ◉ No

(17) * Does your project involve the **collection of any new (or primary) individual or group data** that is **personally or uniquely identifying**? (e.g. data about people or organisations/companies/groups that could be used to identify those individuals or groups; data collection might take any form, including internet and social media data, etc.)

○ Yes ○ No

(18) * Will you ensure that participants who you are collecting data from are provided with **fair warning** and must provide **explicit informed consent** for any data collected?

○ Yes ○ No

(19) * Will you ensure that any project–related data collection, data storage, and data use is in **full compliance** with the **EU General Data Protection Regulation (GDPR)** and the **Data Protection Act (2018)**?

○ Yes ○ No

(20) * Does any of the data that you intend to collect include **sensitive or private personal information** about individuals, or **commercially sensitive information** about organisations/companies/groups?

○ Yes ○ No

21 * Does your project involve **persons under the age of 18 years** (i.e. minors), or **any vulnerable groups**? (e.g. prisoners, refugees, those in care, addiction service users, etc.)

○ Yes ◉ No

22 * Does your project involve the use of **existing (or secondary) human data**? (i.e. data originally collected for another purpose)

◉ Yes ○ No

23 * Is the existing or secondary human data you intend to use either (1) **anonymous/non-personally identifying** and in the **public domain**, or (2) available with **explicit and specific informed consent or permission** for the data to be **legally** reused in the way you intend?

◉ Yes ○ No

24 * Are any aspects of the primary/secondary data you intend to use for the project **controversial** in nature?

○ Yes ◉ No

25 * Before you submit the Ethics Checklist, you must **confirm all of the following**:

☑ I understand that the Ethics Checklist is a formal declaration.
☑ I have answered all questions on the Ethics Checklist carefully and truthfully.
☑ The supervisor/advisor (or principal investigator) for the project is present as the Ethics Checklist is being submitted, or they have given me explicit permission to submit it in their absence.
☑ I have had adequate ethics training and/or instruction prior to completing the Ethics Checklist.
☑ I understand, and agree to abide by, the general ethical principle of "do no harm" for this project.
☑ I will follow the instructions given in the Feedback Report.

26 * Authentication Code (ask your project supervisor/advisor for this code)

Enter Student Number:

Enter the Authentication Code below and click "Verify Code" Verify Code

Note: If an INVALID authentication code is used then this submission is NULL and VOID

6070