2023

Ashar Cosgrove

N11294540

9/11/2023

<your application name>

Replace image with one with some relevance to your application here

**CAB432 Assignment 1**

Contents

[Introduction 2](#_Toc111205932)

[Mashup Purpose & description 2](#_Toc111205933)

[Services used 2](#_Toc111205934)

[Twitter Standard Search API (v.1.1) 2](#_Toc111205935)

[API 2 2](#_Toc111205936)

[API n 2](#_Toc111205937)

[Persistence Service 2](#_Toc111205938)

[Mashup Use Cases and Services 2](#_Toc111205939)

[Restaurant Search in a Foreign City 3](#_Toc111205940)

[US 2 3](#_Toc111205941)

[US n 3](#_Toc111205942)

[Technical breakdown 3](#_Toc111205943)

[Architecture and Data Flow 3](#_Toc111205944)

[Deployment and the Use of Docker 4](#_Toc111205945)

[Test plan 4](#_Toc111205946)

[Difficulties / Exclusions / unresolved & persistent errors 5](#_Toc111205947)

[Extensions (Optional) 5](#_Toc111205948)

[User guide 5](#_Toc111205949)

[Analysis 5](#_Toc111205950)

[Question 1: Vendor Lock-in 5](#_Toc111205951)

[Question 2: Container Deployment 6](#_Toc111205952)

[References 6](#_Toc111205953)

[Appendices 6](#_Toc111205954)

This is a template for your report. It is not compulsory to use it, but it will save a lot of effort if you do. Some of these sections may not be very long, but you should make sure that you cover the key sections describing the application functionality and implementation.

Throughout this document, you should assume that red text in italics is there as guidance and you should read it, follow the instructions, and then **delete it** when you have entered your own text. Some examples are not italicized but should obviously be replaced by your own material. It doesn’t look good when we mark it if the guide text is still there.

There is a mix here between the sections of an ordinary software development report – descriptions of the use cases, technologies used, architecture and testing – and a section in which we ask you to analyse your application based on some prompts that we give you. These two components attract the same number of possible marks.

This report should be around 10-15 pages including screenshots, but this is a guide only – we will not be enforcing a page limit or marking you down for submitting something with 16 pages instead. But be sensible, we really don’t want something that is 25 pages or more.

[Our thanks to the students who allowed us to use images from earlier reports as examples here and in the template for Assignment 2.]

# Executive Summary

This is written in a high-level professional tone. Tell us in no more than a paragraph or so what the app is supposed to do. Explain to us the need for your app and how it provides something that is novel. If there is something especially amazing about your app, tell us briefly what to look for. At this point you can show 1-2 basic screenshots of your application to illustrate the approach, but leave the more detailed screenshotting to the use cases below.

# Introduction

## Description

This is written in a high-level professional tone. Tell us in no more than a paragraph or so what the app is supposed to do. Explain to us the need for your app and how it provides something that is novel. If there is something especially amazing about your app, tell us briefly what to look for. At this point you can show 1-2 basic screenshots of your application to illustrate the approach, but leave the more detailed screenshotting to the use cases below.

The application

## System Requirements

Describe, as user stories, at least two features of your applications. By feature, we mean, the conveniences that your application provides to its user, and not software engineering features. Use the format **As a `<USER-ROLE>` I want the system to `<DO-SOMETHING>` so that `<GOOD-THING-CAN-HAPPEN>`.** For each one of these, you should make it clear how the APIs can support the user story. You do not need to go into extensive detail here, save that for a later section. This is a high-level view of your goals.

User Story 1:

As a streamer, I want to find games similar to what I currently stream so that my audience will be interested in what I'm playing.

The twitch API supports this user story by providing access to the users stream and clip history. This can allow us to determine the most streamed or most popular game, with which we can produce a short list of similar games.

can support this user story by providing a details of the recent or most popular game streamed by the user. Using this to produce a list of similar games with relevant twitch stats.

User Story 2:

As a user, I want to view streams of people playing games similar to what I'm interested in so that I can better gauge whether I will enjoy a given game.

The APIs support this user story by

## Resources

Overview of what API endpoints are used & what they are providing – data sources? Analysis? Here we really do just want a link and a brief description of the API and the services it provides. So, if I used something like Twitter, I would include a description of their service and the endpoint URL. You should repeat this for the others used.

the chosen API services are:

https://dev.twitch.tv/docs/api/ the twitch API provides data about current live streams including what games people are streaming.

https://api-docs.igdb.com/ The IGDB API which provides information about video games such as review scores and short descriptions.

the first user story will utilise the twitch API to get the game most recently streamed game by a user, using the IGDB API to get the games details and a number of games which are similar to it.

the second user story will provide a list of games similar to the one provided along with a stream which is currently live for each game.

## Persistence service

Once you have completed the APIs used for the mashup, you should also list here the persistence service used for the page view counter. For the cloud services, just link to the main page for each service. For Docker, link to the image used.

The persistence service utilised will be the Amazon S3

# Technical Breakdown

This is a deeper discussion of the architecture, the technology used in creating the mashup, any issues encountered, and overall, how you implemented the project.

### Architecture and Data Flow

Explain how your system operates, making it clear how data flows around the system through requests and responses. You are describing the overall architecture of your application at a source code level. The description above tells us something of the application’s use. Now we want to see how that maps to the code organization – show us how your code is organized and tell us how you have split the responsibilities. We should get some sense of how the application works and how the data and control flows around. You may also find it helpful to show us screen grabs of code if that makes your points clearer. Tell us anything you think we need to know about how you have structured the application and made it work, but there also a section below to describe problems.

You are working with a pretty simple mashup, so if your description is more than a page - even including diagrams and code fragments – you are probably giving us more than we need.

### Deployment and the Use of Docker

This section will be relatively brief. You should discuss – usually in a paragraph or two – how you used Docker and any particular configuration choices you want to tell us about. You may include excerpts from the Dockerfile and list this file as an appendix to the report.

If you were unable to get the Dockerfile and Docker deployment working, this is a good place to tell us about those issues.

### Test plan

Manual testing is fine and our expectations are in line with the example grid below. You can show the results through a screen shot and point us to these from the table. If you haven’t used a test plan like this before, then please clarify it with your tutor. We would in general expect 10-15 test cases or so depending on the application.

Your tests should include

* Positive outcome cases
* Negative outcome cases (error scenarios)
* Edge cases
* Non-functional cases (ideally, but not required this time).
* Screenshot evidence that these things actually worked – put these in an appendix

Note that the grid below is unrelated to this application.



Difficulties / Exclusions / unresolved & persistent errors /

In this section, you should explain anything that caused you problems and how you overcame those problems. Tell us if there was any issue that prevented you completing the assignment to specification. Tell us about any assumptions or compromises that you have made. Those who worked with an API like Spotify, which presents particular concerns, should discuss the compromises here, and this is also where you can tell us about problems with API keys and responses.

More generally, you might consider:

* Your major roadblocks and how you resolved them.
* Any functionality you didn’t or couldn’t finish
* Are there any differences between your brief and what you delivered? If so, explain why.
* Are there any outstanding bugs?

Again this section need not be very long or indeed contain much at all – it is more to help us understand your app and to maximise your mark rather than something that we mark directly.

## Extensions (Optional)

In this section, you can tell us if you wish to how you might extend your app and make it better. This is an opportunity to tell us about good ideas that you had that you didn’t have time to tell us about.

## User guide

Tell us how to use your application. You may re-use some of the screenshots from the use case descriptions, but this is more about how to use the app. As long as we can find what we need to do to use your application, this need not be all that long.

But either way, screenshots are your friend.

## Analysis

In this section we ask you look at your application and to analyse it in response to a couple of prompts we supply below. The marking is based on the quality of the analysis and not on the length of your response. There are two questions. In each case, there is an overall question, and then a series of bullet points that help you respond. A good answer can be no more than a couple of sentences in response to each of them. Say more if you have more to say, but don’t waffle. Say it quickly and get on with the next one. This exercise is comparatively straightforward but there is a corresponding task in Assignment 2 which will be far more difficult.

### Question 1: Vendor Lock-in

Looking at your mashup as it stands, how dependent are you on the people who provide the services that you use? In a commercial context, the APIs, the data and the cloud services all matter to you. How hard would it be to change?

In your response, you should consider the following prompts:

How hard would it be to replace the APIs that you are working with? Could you easily replace them if they were shut down suddenly or their terms of service changed? Consider each of them in turn and explore the domain of the API and tell us about obvious competitors or their absence. (4 marks)

How hard would it be to change the persistence service you have used to one supplied by another vendor? Identify equivalent alternatives and discuss briefly how that might affect your approach. (2 marks)

### Question 2: Container Deployment

The mashup you have created as part of this assignment is a very limited application by commercial standards. Without getting too carried away, I want you now to think of a more substantial application which has similar characteristics – drawing mainly on external services, perhaps extending to include some user accounts and security, adding in some additional persistence services like those discussed in the weeks leading up to the assignment submission.

Working with this more substantial application, what are the advantages and disadvantages of the container-based deployment? This new context will in practice involve scaling and load balancing. You don’t yet have direct experience of these services, so I want you to focus on the deployment of the application through software containers. You should refer to earlier notes about the trade-offs between containers and ‘full’ VMs and consider which might apply here, and which might not.

In your response, you should consider the following prompts:

Assuming that we have access to a service to manage container deployment and communication, are there any disadvantages to a container-based deployment for this application? Would we consider deployment of the application directly to a virtual machine i.e. one instance of the application for each instance of a EC2 VM? (2 marks)

Drawing upon the discussions of cloud architectures in the early lectures and the material on persistence from week 5 onwards, what persistence options would you consider if the application were to be deployed at scale using a collection of software containers? You may consider more than one level. (2 marks)

# References

Use a standard approach to referencing – see the guidance at <https://www.citewrite.qut.edu.au/cite/>.

# Appendices

Stuff you want to include but is too long or too complex to include in the main report text. The full Docker file, some longer excerpt from API docs. Whatever helps.