ECM3408 Enterprise Computing
Continuous Assessment 2022-23
Addison Rae's Dinner Party

David Wakeling

Handed Out	Handed In
Thursday 9th February 2023 (T2:04)	Wednesday 22nd March 2023 (T2:10)

This Continuous Assessment is worth 40% of the module mark. Your attention is drawn to Faculty and University guidelines on collaboration and plagiarism.

1 Background

Increasingly, brands collaborate with social media personalities to reach their fan bases, which are often new or hard-to-reach audiences. One such personality is Addison Rae, whose rise to fame began with her posting dance videos on TikTok. By 2021, she was on the Forbes "30 under 30" list for social media influencers, and by 2022, her 88 million followers made her the fourth most-followed individual on the platform.

In the summer of 2022, Google and Samsung sought to leverage Rae's popularity in a YouTube advertisement for their products.

`https://www.youtube.com/watch?v=wbj0s1B_O4o`

This video shows a dinner party at Rae's house. As her bored guests play with their phones, she uses Google Hum-to-Search feature on a Samsung Galaxy phone to identify a song, retrieve it from the Kies music manager, before dramatically casting it to a nearby Bixby-enabled Samsung TV. As soon as the song (which turns out to be "Everybody (Backstreet's Back)" by Backstreet Boys) starts playing, the guests put their phones aside and begin dancing — thanks to Google and Samsung, the party is well and truly started!

This Continuous Assessment is about producing some microservices that can be combined like the Google and Samsung products.

Part 1: A Tracks Microservice (40%)

Show some Go code for a *Tracks* microservice that, like *Kies*, stores music tracks in an SQL database. This service should listen on port 3000.

Creating Tracks

Tracks allows music tracks to be created. See Figure 1. Here, the `PUT` method takes

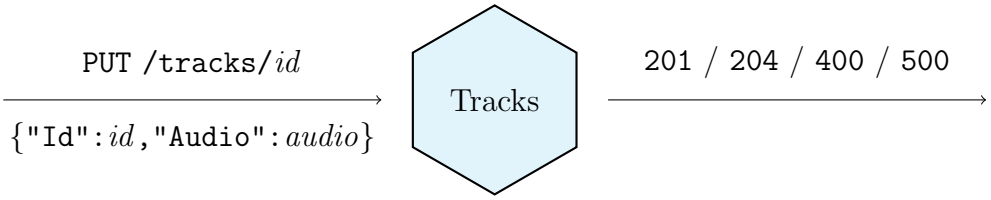

Figure 1: Creating tracks.

a JSON object that has an `Id` property whose value is a string, and an `Audio` property whose value is a WAV file encoded using the Base64 scheme. Successful results are `201 Created` and `204 No Content`. Unsuccessful results are `400 Bad Request` and `500 Internal Server Error`.

This `PUT` method may be tested with the script

```
#!/bin/sh
ID="Everybody+(Backstreet's+Back)+(Radio+Edit)"
ESCAPED=`perl -e "use URI::Escape; print uri_escape(\"$ID\")"`
AUDIO=`base64 -i "$ID".wav`
RESOURCE=localhost:3000/tracks/$ESCAPED
echo "{ \"Id\":\"$ID\", \"Audio\":\"$AUDIO\" }" > input
curl -v -X PUT -d @input $RESOURCE
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	1	2	3	4	5	6	7	8	9	10
mark											

Listing Tracks

Tracks allows all music tracks to be listed. See Figure 2. A successful result is `200 OK` accompanied by a list of strings. An unsuccessful result is `500 Internal Server Error`.

Figure 2: Listing tracks.

This `GET` method may be tested with the script

```
#!/bin/sh
RESOURCE=localhost:3000/tracks
curl -v -X GET $RESOURCE
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	1	2	3	4	5	6	7	8	9	10
mark											

Reading Tracks

Tracks allows a music track to be read. See Figure 3. A successful result is `200 OK`

Figure 3: Reading tracks.

accompanied by a JSON object that has an `Id` property whose value is a string, and an `Audio` property whose value is a WAV file encoded using the Base64 scheme. Unsuccessful results are `404 Not Found` and `500 Internal Server Error`.

This `GET` method may be tested with the script

```
#!/bin/sh
ID="Everybody+(Backstreet's+Back)+(Radio+Edit)"
ESCAPED=`perl -e "use URI::Escape; print uri_escape(\"$ID\")"`
RESOURCE=localhost:3000/tracks/$ESCAPED
curl -v -X GET $RESOURCE
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	1	2	3	4	5	6	7	8	9	10
mark											

Deleting Tracks

The Tracks microservice allows a music track to be deleted. See Figure 4. A successful

Figure 4: Deleting tracks.

result is `204 No Content`. Unsuccessful results are `404 Not Found` and `500 Internal Server Error`.

This `DELETE` method may be tested with the script

```
#!/bin/sh
ID="Everybody+(Backstreet's+Back)+(Radio+Edit)"
ESCAPED=`perl -e "use URI::Escape; print uri_escape(\"$ID\")"`
RESOURCE=localhost:3000/tracks/$ESCAPED
curl -v -X DELETE $RESOURCE
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	1	2	3	4	5	6	7	8	9	10
mark											

Part 2: A Search Microservice (40%)

Show some Go code for a *Search* microservice that, like *Hum-to-Search*, provides music recognition. This service should listen on port 3001.

Search allows music fragments to be recognised. See Figure 5. Here, the `POST` method takes a JSON object that has an `Audio` property whose value is a WAV file encoded using

Figure 5: Recognising tracks.

the Base64 scheme. A successful result is `200 OK` accompanied by a JSON object that has an `Id` property whose value is a string. Unsuccessful results are `400 Bad Request`, `404 Not Found` and or `500 Internal Server Error`.

This `POST` method may be tested with the script

```
#!/bin/sh
ID="~Everybody+(Backstreet's+Back)+(Radio+Edit)"
AUDIO=`base64 -i "$ID".wav`
RESOURCE=localhost:3001/search
echo "{ \"Audio\":\"$AUDIO\" }" > input
curl -v -X POST -d @input $RESOURCE
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	4	8	12	16	20	24	28	32	36	40
mark											

To keep the implementation of this *Search* microservice manageable, it should be based on the service for recognising fragments in files described at "`https://audd.io`".

Part 3: A CoolTown Microservice (20%)

Show some Go code for a *CoolTown* microservice that, like *Bixby*, provides for device integration. This service should listen on port 3002.

CoolTown allows music tracks to be retrieved using music fragments. See Figure 6. Here, the `POST` method takes a JSON object that has an `Audio` property whose value is a WAV file encoded using the Base64 scheme. This audio is a music fragment. A successful result is `200 OK` accompanied by a JSON object that has an `Audio` property, whose value is a WAV file encoded using the Base64 scheme. This audio is a music track. Unsuccessful results are `400 Bad Request`, `404 Not Found` and `500 Internal Server Error`.

This `POST` method may be tested with the script

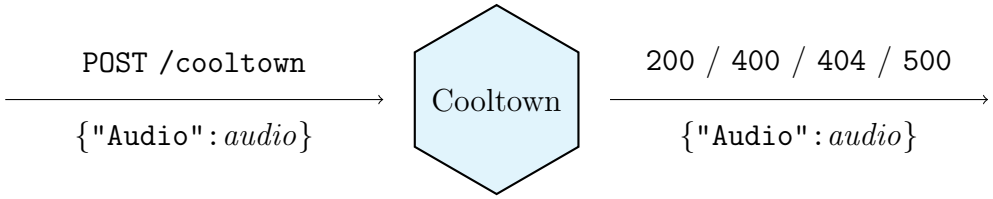

Figure 6: Retrieving tracks.

```
#!/bin/sh
ID="~Everybody+(Backstreet's+Back)+(Radio+Edit)"
AUDIO=`base64 -i "$ID".wav`
URL=localhost:3002/cooltown
echo "{ \"Audio\":\"$AUDIO\" }" > input
curl -v -X POST -d @input $URL
```

Your work will be assessed automatically using scripts designed to produce each possible result.

	0	2	4	6	8	10	12	14	16	18	20
mark											

2 Submission

You should submit a single ".zip" file (note, *not* 7zip or WinRAR) containing the Go source code of your microservices in the directories "`addison/tracks`", "`addison/search`" and "`addison/cooltown`" by midday on *Wednesday 22nd March 2023*.