University of Exeter

COURSEWORK SPECIFICATION
ECM3408 — Enterprise Computing
Module Leader: David Wakeling
Academic Year: 2024/25

Title: **Coursework Assignment**

Submission deadline: **Wednesday 5th March, 2025, Midday**

This assessment contributes **40%** of the total module mark and assesses the following **Intended Learning Outcomes**:

- Demonstrate recognition of the problems that can arise in the development of large-scale distributed information systems;
- construct concurrent and distributed computing systems using an enterprise-level model (e.g. EJB, .Net);
- understand and use protocol specifications;
- design and implement heterogeneous systems.

This is *an individual* assessment and you are reminded of the University's Regulations on Collaboration and Plagiarism. You must avoid plagiarism, collusion or any other academic misconduct. Further details about Academic Honesty and Plagiarism can be found at `https://ele.exeter.ac.uk/course/view.php?id=1957`.

Background

Shazam is a service that can identify a music tracks from music fragments, provided that the music track is present in the system's database. The service was launched in the UK as "2580," the short code that customers dialled on their mobile phone to get a music fragment recognized. Originally, the customer sent a text message containing a music fragment, and was sent back a text message containing music track title and artist name. Later, the service allowed the music track to be downloaded too.

Specification

This continuous assessment is to build a Shazam-like MVP called Shamzam using some microservices. The MVP specification is described by four user stories.

S1 As an *administrator*, I want to *add a music track to the catalogue*, so that *a user can listen to it.*

S2 As an *administrator*, I want to *remove a music track from the catalogue*, so that *a user cannot listen to it.*

S3 As an *administrator*, I want to *list the names of the music tracks in the catalogue*, so that *I know what it contains.*

S4 As a *user*, I want to *convert a music fragment to a music track in the catalogue*, so that *I can listen to it.*

Assessment

Design (20%)

Show a collection of microservices that may be used implement Shamzam user stories S1, ... S4. For each microservice, there should be a one or more endpoint diagrams. See Figure 1. These diagrams must show the Representational State Transfer (REST) API

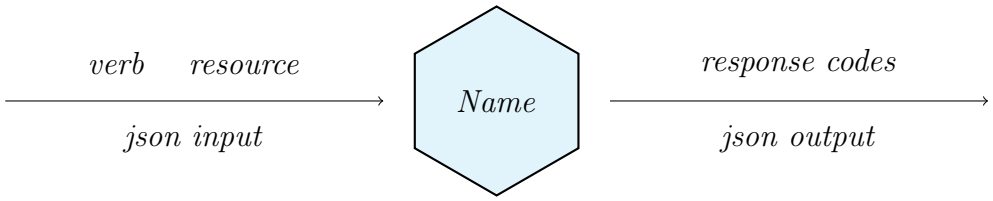

Figure 1: A microservice endpoint diagram.

that the endpoint provides.

Marks will be awarded for:

- *effectiveness* — does the collection of microservices fully implement Shamzam user stories S1, ... S4? (10%)
- *elegance* — is the collection of microservices highly cohesive, loosely coupled, and just sufficient to implement Shamzam user stories S1, ... S4? (10%)

Implementation (60%)

Show some Python modules that implement your microservices. This code must implement the REST API that each endpoint provides. Any database storage needed should be implemented using an SQLite database. Any audio recognition needed should be implemented using Audd.io.

Marks will be awarded for:

- *clarity* — are statements, expressions and functions used appropriately to a give well-structured programs? (30%)
- *correctness* — are inputs processed correctly to give outputs, including the signalling of invalid input, errors in processing or failures of communication? (30%)

Testing (20%)

Show some Python code that implements unit tests (technically, end-to-end tests) for each Shamzam user story.

Marks will be awarded for:

- *coverage* — do unit tests cover "happy paths" and upto three "unhappy paths" (20%)

Useful Materials

A number of useful materials have been made available on the ELE server to assist you in completing the assessment:

- `ENV.txt` — an AudD.io key, good for 20,000 fragment lookups;
- `~Blinding Lights.wav` — a fragment of *Blinding Lights*, by The Weeknd;
- `~Don't Look Back In Anger.wav` — a fragment of *Don't Look Back In Anger*, by Oasis;
- `~Everybody (Backstreet's Back) (Radio Edit).wav` — a fragment of *Everybody (Backstreet's Back)*, by Backstreet Boys;
- `~good 4 u.wav` — a fragment of *good for u*, by Olivia Rodrigo;
- `~Davos.wav` — a fragment of a speech to the World Economic Forum, by Donald Trump;

- `Blinding Lights.wav` — (some of) the track *Blinding Lights*, by The Weeknd;
- `Don't Look Back In Anger.wav` — (some of) the track *Don't Look Back In Anger*, by Oasis;
- `Everybody (Backstreet's Back) (Radio Edit).wav` — (some of) the track *Everybody (Backstreet's Back)*, by Backstreet Boys;
- `good 4 u.wav` — (some of) the track *good for u*, by Olivia Rodrigo.

AI-supported

This module is *AI-supported*. That is, it is one

> *where ethical and responsible use of GenAI tools in the development of an assessment is supported. This may include using GenAI tools to summarise literature, improve the structure of your work or quality of English language. All use of GenAI tools should be acknowledged in a statement submitted with their assessment and referenced appropriately. Students are asked to keep a record of the tools, prompts and outputs used so they are able to produce these if necessary at a viva and demonstrate how they have built on this content to ensure the work is original.*

Submission

You should submit a single ".zip" file (note, *not* tar, 7zip or WinRAR) containing a Generative AI Declaration, as a ".pdf" file, saying how (if at all) Generative AI was used in the submission, a microservices design, as a ".pdf" file, and the Python code of your microservices and unit tests in separate ".py" files. Other Python source code files may be supplied as needed. Your Python source code must run on the Anaconda implementation in the Lovelace laboratory. Should any question arise as to whether code runs or not, it will be answered on the Lovelace implementation.