**University of Stirling  
Computing Science and mathematics  
CSCU9YW Web Services Assignment**

**Autumn 2022**

**Module coordinator: Dr Patrick Maier**

This assignment will assess your understanding of web services and your skill in designing and implementing an application based on RESTful web services and clients that interact with these services.

**TASK**

*Build a web-based polling application.* The application shall comprise of a web service following REST design principles, and of clients that interact with the web service. The web service must be implemented in Java as a Spring Boot application. You are free to implement the clients using whatever languages or frameworks you prefer.

*Demonstrate your application* and record the demonstration in a screencast video. The video should be no more than 10 minutes long. Make sure that text on screen is readable in the video.

*Write a technical report on the polling application.* Briefly discuss the problem, any assumptions you made, and what functionality you have managed to get working. Then discuss your solution in detail. Focus mainly on the web service, discussing

* what resources it manages and how they are represented,
* what data structures (or data base schema) it uses to store resources,
* how and to what extent your solution follows the REST design principles, and
* how and to what extent your solution ensures scalability, reliability and security.

The report should be *critical and reflective* in nature; you may want to discuss alternative approaches to implementing certain functionality, or justify some of your implementation choices (e.g. the choice of data structure or data base schema). You may illustrate your report with diagrams or screenshots where appropriate. Make sure the report is appropriately formatted and structured as a professional technical document, and that it includes a cover page stating the module code and your student number (but not your name).

Include full listings of your source code (of the web service and of the clients) as appendices in the report. Make sure that your source code is formatted properly, e.g. use a monospace font. You can reduce the font size for the appendices, e.g. to 8 pt. **Do not** include screenshot images of your code in the appendices.

*Further information:* A specification of the functionality of the polling application, instructions how to demonstrate the application, and assessment criteria for the assignment are provided at the end of this document.

## Submission Arrangements

You must submit your solution to the *Coursework* assignment on Canvas before **23:59 on 21 November 2022**. Your submission must consist of three files:

1. a **ZIP** archive containing all your code (web service and clients),
2. the screencast recording of your demonstration (in **MP4** format), and
3. the report (in **PDF** format).

All three parts are required; submissions that are missing one of the required files will be regarded as non-submissions. Please upload three separate files; do not include the PDF report in the ZIP archive, as the report must be processed by Turnitin.

Make sure that you do not reveal your name in your source files. Instead, include your student number as a comment in each file to mark authorship.

## Late Submission

If you cannot meet the assignment deadline and have good cause, please apply for an extension on Canvas.

Coursework will be accepted up to seven calendar days after the submission deadline (or expiry of any agreed extension) but the mark will be lowered by three percent per day or part thereof. After seven calendar days, the piece of work will be deemed a non-submission and will be awarded zero marks.

## Note on Avoiding Academic Misconduct

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on Academic Integrity and Academic Misconduct (including plagiarism) which can be found at <https://bit.ly/37fYxPw>.

*Plagiarism*: We are aware that assignment solutions by previous students can sometimes be found posted on GitHub or other public repositories. Do not be tempted to include any such code in your submission. Using code that is not your own will be treated as “poor academic practice” or “plagiarism” and will be penalized. To avoid the risk of your own work being plagiarised by others, do not share copies of your solution, and keep your work secure both during and after the assignment period.

*Collusion*: This is an individual assignment: working together with other students is not permitted. If students submit the same, or very similar work, this will be treated as "collusion" and all students involved will be penalized.

*Contract cheating*: Asking or paying someone else to do assignment work for you (contract cheating) is considered gross academic misconduct, and will result in termination of your studies with no award.

# *Polling service*

**Specification of web service and client functionality**

The *Scottish Association for the Watching of Birds (SAWB)* conducts an annual poll of its members to elect the Bird of the Year. Your task is to build a web service for this poll.

The polling service’s main function is to let SAWB members cast ballots for Bird of the Year candidates. Ballots are not anonymous, that is, the service records how individual members vote. However, ballots are secret, that is, members can inspect their own ballots but they cannot find out how anyone else voted.

The polling service offers additional functionality to administrators, who can open and close the poll, and tally the vote. However, administrators cannot inspect individual ballots because the polling service protects the secrecy of ballots.

The polling service should store information about Bird of the Year candidates, SAWB members and their ballots in suitable Java data structures, or in a data base. The information about Bird of the Year candidates includes their common name (e.g. raven), their scientific name (e.g. corvus corax), and a description. The information about SAWB members includes their unique membership number, their name, their age, and the region of Scotland where they live.

The list of Bird of the Year candidates may be hard-coded into the polling service; the poll must offer a choice between at least three candidate birds. Information about SAWB members is available from a membership register. This is a separate web service that maintains an evolving data base of synthetic membership records. Membership changes over time; some people cease to be members while others newly join the SAWB. Details on how to retrieve information about members from the membership register will be published on Canvas. You should use this membership register when testing your web service.

**SAWB member view of the polling service**

The polling service should let SAWB members list all Bird of the Year candidates, cast a ballot for a candidate, retract their ballot, and inspect which bird they have voted for.

In order to cast a ballot, SAWB members must identify themselves to the polling service by providing the information stored by the membership register (membership number, name, age and region of Scotland).

Every member has one vote, but they are allowed to change their opinion and cast ballots multiple times – only the last ballot counts. Members can also retract their ballot, which has the same effect as if they never voted.

Optionally, the polling service may verify a voter’s eligibility to vote by checking their information against their record on the membership register. Ballots from ineligible voters should be rejected.

You may create a polling client as a user interface for SAWB members, implementing the request/response interactions with the polling service. (If no such client is built, you can use generic clients such as curl or Postman for testing the polling service.)

**Admin view of the polling service**

The polling service should let administrators tally the votes cast and list the tally for each Bird of the Year candidate. The remainder of the admin functionality is optional.

The polling service may let administrators open and close the poll. If this is implemented then casting or retracting of ballots should only be possible after the poll opens and before it closes.

Access to the polling service’s admin functionality should be secured appropriately so that only legitimately authenticated administrators can access the functionality.

You may create an admin client as a user interface for administrators, implementing the request/response interactions with the polling service, including authentication.

**Demonstration of the polling application**

Your screencast video must demonstrate all functionality that your report claims to be working. To demonstrate the functionality, your video must show (at least) two windows simultaneously:

* A window displaying the client. This may show your polling or admin client, or a generic client like Postman, or curl running in a command prompt or shell.
* A window displaying the console output of the web service.

You must implement your web service in such a way that the console output contains enough information to infer what requests the service is receiving. As a minimum, crucial HTTP header information (e.g. the HTTP method, the URL path and the response status code) of *valid* requests should be logged.

To prepare for the demonstration, start up a polling service with at least 3 candidate birds. Retrieve at least 3 random SAWB members from the membership register.

Start the screencast recording and use the 3 random SAWB members to demonstrate the functionality from the members’ point of view: Listing all candidate birds, casting ballots for different candidates, retracting ballots and inspecting a member’s own vote. To demonstrate the admin functionality, tally the votes. In addition, demonstrate any optional functionality that you have implemented, e.g. voter eligibility checks, or opening/closing the poll.

While demonstrating, explain the effect of each request on the polling service. Choose your sequence of requests in such a way that the effects can be observed in the video.Include potential error cases, e.g. casting a ballot for a bird that isn’t a Bird of the Year candidate.

**Assessment Criteria**

This assignment is worth **100%** of the overall mark for the CSCU9YW module.

The mark for the assignment depends on how much of the functionality was implemented and demonstrated, on the quality of the code, and on the quality of report and the demonstration. The code and the demo account for 60%, the report for the remaining 40% of the mark for this assignment.

Your report is assessed according to how clearly and comprehensively it discusses your solution.

Your demonstration is assessed according to how well it shows the behaviour of your solution, including the behaviour in error cases.

Your code is assessed according to the following criteria:

* Correctness: Are the aspects that were implemented working as specified?
* Completeness: How much of the specified functionality is working?
* Complexity: How sophisticated is the solution, in terms of data handling, error handling, following REST principles, and appropriate use of programming constructs?
* Documentation: How well documented is the code, e.g. are comments used appropriately and intelligently?
* Structure: How well structured is the code, e.g. is it suitably modularised, and is the layout consistent?

Correctness and completeness are the two most important criteria. Provided your report and demo are decent, you can achieve a pass for this assignment while omitting all optional features. That is, for a pass your polling service much let SAWB members vote, retract and inspect their ballot, and the service must tally the votes. It is sufficient to demonstrate this functionality using generic clients like Postman or curl – you do not need to implement the polling and admin clients for a pass.

To achieve a first for this assignment, you will have to write an excellent report and implement and demonstrate almost all of the optional functionality, including polling and admin clients. In addition, your code must be judged as excellent with regard to the remaining code assessment criteria, particularly complexity.

Higher marks are available for solutions which implement a polling application following HATEOAS principles, or which excel in securing admin access.