Scalability is a very important aspect that a webservice must provide. In this project, many measures were taken to increase scalability.

One important aspect of the REST-based web service is that it is stateless, this means that any service replica can service any client. Due to this, it was most appropriate to use a resource per request approach to deal with incoming requests. This is a very scalable approach as if traffic increases, we can have additional service replicas to accommodate.

One issue that had to be dealt was with authentication, since the service was stateless, it should not store any information about clients. Therefore, for authentication, we must use a token-based authentication system. A random universal unique identifier (UUID) is granted to a user upon authentication, this is a 128-bit value which will uniquely identify users. The client must supply this UUID token in the form of a cookie when making other requests to the service. UUID is used as there is a very miniscule chance of two users being granted the same UUID meaning our service can support many users aiding in scalability.

Optimistic locking was another feature that was used to aid in the scalability of this project. Users can concurrently access and book seats, however, due to the way the optimistic lock is implemented, two different users are not able to book the same seat. This is a far more scalable solution to the issue of supporting the reliability constraint of only one user being able to book a seat for a given concert on a given date.

Pessimistic locking is a suboptimal solution to this constraint as it means that only one user will be able to access and book seats at a time, this does not support concurrent access and will vastly decrease server throughput.

To reduce server load, the client has a cache of concerts and performers. These are used in the GET methods for getting all concerts and getting all performers when multiple requests are done in a short amount of time. The server will only need to send out a required information once and for subsequent requests in the timeframe, a not-modified message will be sent which will prompt the client to search in their local cache. Caching has benefits in that the service will not need to continuously go through the process of loading and sending all the objects, this will improve the throughput of the service.

Image loading is another aspect in which local storing will benefit scalability. Images are only downloaded to a client once; subsequent retrievals will be made from local storage. This will reduce load on the server as well as reducing the amount of data downloaded by the client.

As a whole, the service is reasonably scalable, due to the measures taken, however there may be some concerns with server memory as concert number grows as the entity manager is not cleared intermittently.