**Restful services (Representational State Transfer)**

**Django**

Django is a python based lightweight web service that contains an SQL lite database, the framework uses a MVC (Model View Controller) pattern to organise the structure of the application. With Django’s rest framework a client can send simple http request over to the server and get fast responses in the form of JSON. The advantages of Django’s lightweight structure is that it has a very high performance time and can be efficiently scaled. There are several security mechanisms Django has such as the management of roles and groups of users and there access rights, the use of SSL/HTTPS and protection again several forms of common web attacks (e.g, clickjacking, Cross site scripting and cross site request forgery). The other two forms of protection are different then any of the other service frameworks I have chosen and feel this has had a large impact on the decision as these would be important to our project, these are header validation so that http requests are not malicious and sql injection projection, this is because Django won’t allow the developer or client create raw sql requests and instead holds by an object orientated concept where the service creates an object that is read as an insert request and editing or deleting records from the database is done by calling that object.

The down side to Django is that with comparisons to other database focused relational mapping tools like SQLAlchemy there is is less functionality when it comes to manipulating data in the database. The code base when scaled can be large and hard to organised for example if the project was a large web application with several views and features then it would be hard to organise the file structure. Finally although the framework and the python language behind it is simple some of the technical concepts behind the system are quite difficult to understand and therefore to create an effective and efficient system the developer needs to understand the whole system.

**Spring**

Spring is similar framework to Django is the way that its infrastructure is contained with in a wrapper so that the developer doesn’t have to organise it themselves, however the it is different interns of the language it is built on (Java) and the pattern of dependancy inject it is made on which is where instead of the object being created for the dependancy, the dependancy can be called as a property setter within the constructor.

The advantages spring has over Django is that the system is more customisable when its comes to what libraries to use, which database to use and which containers to use. Where Django has a strict sql lite database it utilises and the user can’t utilise raw sql code. Spring has free reign on which data base it can use based on the systems needs and the developer can decide to use libraries like JDBC to query the database with sql.

The downside to spring however is which the increase of customisation comes the risk or vulnerabilities which means the developer has to validate and sanitise the sql for each request to the database. The libraries that the framework depends can be large and difficult to understand. Finally the documentation for the framework is quite sparse in areas and would be difficult to learn advanced concepts dues to little documentation.

**Play**

Play is another java framework with very similar concepts to Django and spring, also using the MVC pattern, however it is built using several technologies, the dependancy injection that it uses it made with guice, and the rest foundation is build on Akka which is used to create concurrent applications. Similar to Django it is a very lightweight framework that focuses on minimising the consumption of performance with CPU and memory.

The advantages play has over the other frameworks is that as it has been built on several concurrent technologies such as akka and nettie, it means that it is very inexpensive to deal with remote calls in parallel which would be important for our project as the service would be potentially called by several users at the same time.

The disadvantages of play however are that similar to spring it lacks good documentation which makes its difficult for new users to understand the framework, the build system of play (SBT) is good for a lot of interesting ways such as hot reloading however the system has a very large code base and is not documented in a way that even developers who understand the application can get to grips with.

**Baas (Backend As A Service)**

**Game Spark**

Finally a framework that is very different from all the frameworks i have discussed which is a BAAS (Backend As A Service), this operated as a cloud service which aims to provide developers with a pre-template API service that the client can call, this makes it easier for developers to set up the backend and focus more on the front end of the system. Game spark is a gaming orientated BAAS and comes with numerous features and functionality which would be advantageous to our project, such as session management for multiplayer games, accolade and leaderboard setup and levels of security that manage player roles and accounts. This style of service has an everything in the box concept where the developer doesn’t need to create anything they just use pre existing templates.

With communication to the front end it accepts the restful concepts that we have discussed previously. However these are commercially provided and come at a price for projects, this due to the resource constraints we have means that we won’t be using this service.