Period-2 Exam Questions and notes:

**Explain Pros & Cons in using Node.js + Express to implement your Backend compared to a strategy using, for example, Java/JAX-RS/Tomcat**

***Pros:*** *We can easily build the frontend in the same language, and having the same language will make for easier communication between the two ends. With Node and Express, it is easy to scale the project up if needed compared to other languages. Other then that, there are both a lot of community support and development here, as well as it being supported by the Google V8 engine, a very strong platform among the current browsers. With all this support, there is also a lot of third party applications that can be added to many projects.*

***Cons:*** *Because of single-threading and the needs for the event-loop and callbacks, there are potential issues here. Middleware can be strong, put have some complexity to solve different issues, compared to a language like Java, and can take a long time to fix in development.*

**Explain the difference between *Debug outputs* and *ApplicationLogging*. What’s wrong with console.log(..) statements in our backend code.**

*With Debug, we are able to print values and such while only in the development phase of the running code, and can avoid having them show up while the code is in production. Beyond that, console.log() might add to the time each code execution takes. In short, debug is like a console.log that only print when we want it, and does force us to ‘clean up’ the code in the same way before it can go into production*

**Demonstrate a system using application logging and environment controlled debug statements.**

***Week7***

**Explain, using relevant examples, concepts related to testing a REST-API using Node/JavaScript/Typescript + relevant packages**

***Week9, userEndpointDBTest.ts***

*Here we are using the Mocha Testing Framework, and Chai Assertion Library, in order to test our code, as well as utilizing async/await to handle the promises from accessing the REST-api.*

**Explain a setup for Express/Node/Test/Mongo-DB development with Typescript, and how it handles "secret values", debug and testing.**

***Week9***

***.env:*** *For the secret values, we can store them in environmental variables, or in a .env file.*

***package.json:*** *Here we can specify how to convert typescript files into js for production, and specify which testing framework to use and where, among many other system specifications.*

***Testing:*** *We use a testing framework like* ***Mocha****, and utilizes an assertion library like* ***Chai***

***.ts:*** *By using* ***Typescript*** *we can use types for the coding, supporting the understanding thought development about what values are send where and make sure we know what we are using.*

**Explain, preferably using an example, how you have deployed your node/Express applications, and which of the Express Production best practices you have followed.**

*As of right now, I’m using a docker container solution for my upload, and github itself as my pipeline with secret values given, to help specify which docker images to utilize.*

**Explain possible steps to deploy many node/Express servers on the same droplet, how to deploy the code and how to ensure servers will continue to operate, even after a droplet restart.**

*With my current setup, the steps can be followed from these links,* [*https://github.com/AlexanderSarson/sem4\_devops\_docker*](https://github.com/AlexanderSarson/sem4_devops_docker)*,* [*https://github.com/AlexanderSarson/sem4\_devops\_ci\_backend\_nodejs*](https://github.com/AlexanderSarson/sem4_devops_ci_backend_nodejs)

**Explain, your chosen strategy to deploy a Node/Express application including how to solve the following deployment problems:**

* **Ensure that you Node-process restarts after a (potential) exception that closed the application**
* **Ensure that you Node-process restarts after a server (Ubuntu) restart**
* **Ensure that you can run “many” node-applications on a single droplet on the same port (80)**

***Needs to be added later!***

**Explain, using relevant examples, the Express concept; middleware.**

***Week7***

*Middleware are used for Express, in order to add a single/sequence of function that a request will pass thought as it is being received. The order of how the middleware is applied in the code (using express().use), determines the order of how the middleware is used. Middleware are function that take 3 parameters, the request, response and next, that will move from the current middleware to the next.*

**Explain, using relevant examples, your strategy for implementing a REST-API with Node/Express  + TypeScript and demonstrate how you have tested the API.**

***Week9***

*By using Express.Router(), we are able to add the endpoint for when we use Express.Listen(Port) on a given port. We can then test this, by changing the port values in the .env file into one for testing, changing the other hidden variables so we don’t have to worry about authentication when we don’t want to and so on, and then test on it.*

**Explain, using relevant examples, how to test JavaScript/Typescript Backend Code, relevant packages (Mocha, Chai etc.) and how to test asynchronous code.**

***Week9***

*By using assertions from the Chai Library and framework from Mocha module, we are build a test-suit in a given test file, and point to that file in the package.json file. From here, we can expect the results we want, inside a describe method callback. For async code, we can either make the callbacks themselves async, or we can* ***expect(value).to.eventually.have.property*** *from Chai as Promised.*

NoSQL and MongoDB

***Explain*, generally, what is meant by a NoSQL database.**

*A NoSQL is meant to simply not be a SQL database, and as such, data inserted into a NoSQL database is not required to meet the same specific specification as in a traditional SQL database. As such, a Person object could be stored with different values from each other, or a Phone object as well, all in the same Table, which is here referred to as a Collection. Each inserting of a Row, here called Document, don’t have to have the same specifications.*

***Explain* Pros & Cons in using a NoSQL database like MongoDB as your data store, compared to a traditional Relational SQL Database like MySQL.**

*The pros of using a NoSql database is that it is more easily scalable by using a horizontal scalability, allow multiple machines to work together to increase the strength of the database. With NoSQL, data doesn’t have to be strictly formatted, and can more easily have added of missing values compared between documents in a single collection. Some of the cons are about how the lack of structure for the data uploaded will require more code to handle, as there is no guarantee about the condition or format of the data, as it is being pulled from the database.*

***Explain* about indexes in MongoDB, how to create them, and *demonstrate* how you have used them.**

***Week11***

*The indexes in MongoDB allows for faster searches for specific data in the database, and can be specified to single or multiple values. These indexes will be sorted when searching upon them, and if there is multiple, by the values in order.*

***Explain*, *using your own code* examples, how you have used some of MongoDB's "special" indexes like *TTL* and *2dsphere and perhaps also the Unique Index.***

***Week11***

*The TTL indexes stand for ‘Time Too Live’, and the database notes when the data was added, and makes the data invalid/removes data, that has passed it’s expiration time.*

*2dsphere indexes can be used for geograthic 2d coordinates, and can be used to find indexes within a certain area quickly.*

*Unique index can be given, like usernames, to quickly find specific users.*

***Demonstrate*, using a REST-API *designed by you*, how to perform all CRUD operations on a MongoDB**

***Week11***

*CRUD, standing for Create, Read, Update, Delete, is about how to handle and manipulate data in the MongoDB with a DB, from different Collections, upon any/all documents within any collection.*

***Explain*, using a *relevant example*, a full JavaScript backend including relevant test cases to test the REST-API (not on the production database)**

***Week9, test files***

*To test for the mongoDB all the way from our own REST-api, we need to set up a testing port to test from, and preferably communicate with a test Collection within our MongoDB, so we don’t affect any real data.*

**Demonstrate, using your own code-samples, decisions you have made regarding → normalization vs denormalization**

***Not added yet***

Geo-location and Geojson

**Explain and demonstrate basic Geo-JSON, involving as a minimum, Points and Polygons**

***Week10***

*Basic GeoJson needs to be set up in a specific formate, using a type string to identify the type, and an array or coordinates, depending on the type of format the Geo-Json-opject is in. The ordering of longtitude and latitude needs to be in a speficiec order, which is often the reverse of what is given in api’s.*

**Explain and demonstrate ways to create Geo-JSON test data**

*An online tool such as* [*http://geojson.io*](http://geojson.io/) *can be used, or it can be written manually if the formatting of the json code is know, which can be seen at a place like here:* [*https://github.com/maxogden/geojson-js-utils*](https://github.com/maxogden/geojson-js-utils)*.*

**Explain the typical order of longitude and latitude used by Server-Side APIs and Client-Side APIs**

*The Geo-Json we utilize uses longitude latitude, which is the way it needs to read the data, but because of things like data-tradition and ongoing utility, most api’s uses the reverse latitude longitude. The reason this is important to remember, is to save time from annoying debugging, when coordinates can unfortunately easily be confused and switched around.*

**Explain and demonstrate a REST API that implements geo-features, using a relevant geo-library and plain JavaScript**

***Week11***

*By making sure coordinates is send, the order also has to be correct, and as such when just receiving a single point, values can be named so to not be switched around.*

**Explain and demonstrate a REST API that implements geo-features, using Mongodb’s geospatial queries and indexes.**

***Week11***

**Explain and demonstrate how you have tested the gameFacade and gameAPI for the game-related parts of the period exercises**

**Week11**