* Create ‘.gitignore’ file in vs code and ad ‘node\_modules/’ node modules folder wont be added to the repository
* Initialize repository, ‘git init’
* Initialize package.json (npm init)
* Installing regular dependencies: npm I

1. express (main web framework for the backend)
2. express-validator
3. bcryptjs (used for password encryption)
4. config (package for global variables)
5. gravatar (for profile avatars)
6. jsonwebtoken
7. mongoose
8. request (a small module that allows us to make http request to another api

* Installing dev dependencies: npm I -D

1. Nodemon (constantly watches server for changes)
2. Concurrently (allows us to run our back end express server and our front end react dev server at the same time with single command )

* We need to have main entry file, we’re creating server.js
* In package.json, replace test with “start” : “node server” that’s the script that Heroku will run when we deploy
* …for development we create script named “server”: “nodemon server” so that we don’t have to keep refreshing the server.
* Npm run server, send get request on postman to get response.
* Commiting: (git add .), (git commit -m ‘Initial commit’ ).
* Now connecting database. On mongo db site, click connect, connect your application, copy the string.
* Create new folder called config, in that folder create a file ‘default.json’. That file will have our default global values.. we need to create one for mongo db string, whats called mongo URI.
* Inside config create another file ‘db.js’, here we do our mongodb connection.
* Bring in connectDB in server.js
* ROUTES…create new folder ‘routes’, in routes new folder ‘api’
* In api create:

1. Users.js – that will handle registering users, adding users…
2. Auth.js – which will handle getting json webtoken for authentication
3. Profile.js – which will have routes that have anything to do with profiles, fetching them, adding them, updating them…
4. Posts.js – we will have a little form area where we can add posts, like, comment etc.

* Users.js – we need to bring in express router if we’re going to have routes in separated files like this.

Public/Private - If we need a token to access a specific route like adding a profile, we need to be authenticated, we need to send a token to that route in order to work, otherwise we get something like unauthorized access. The way we’re gonna make that happen is by creating our own auth middleware later on.

* In server js we gonna make it so that we can actually access the routes.
* In postman we check if the reoutes work: <http://localhost:5000/pai/profile>...
* Within postman, we wanna separate our resources for our endpoints into collections.
* **Creating the user model:** create new folder in root ‘models’ and inside a file ‘Users.js’. To create a model, we need to create a schema which just holds the different fields that we want this particular resource to have.
* **Request & body validation** bring in express validator in users.js
* After getting JWT back with user payload, we now need to make it so that we can send that token back to authenticate and access protected routes. The way we’re gonna do that is creating our own custom middleware. Create folder in the root ‘middleware’. Middleware/auth.js. After that we implement it in protected route.
* Routes/Api/auth.js --- bring in the middleware.
* Then we can check in postman if the route is protected. Send a get request to <http://localhost:5000/api/auth> , with x-auth-token and value is the token that we got.
* Routes/api/auth.js - …
* Now we want to make it so we can actually log in. Right now we can register, it gives us a token back, but we wana be able to take the credentials of the user, the email and password and send that to a route.
* Routes/api/auth - Bring in express validator, config and jwt.
* **Creating the profile model**  - models/ make new file ‘Profile.js’
* Now into orutes/api/profile.js. We need to create route to get profile.
* **Create and update profile:**  orutes/profile.js –bring in express validator
* **Create route to get all profiles**
* **Delete profile an duser**
* **Get Github Repos for profile –** github.com/settings/developers, register new OAuth app. Generate client secret. Go to default.json and add: “githubClientId” and “githubSecret”. Now back to profile.js route file. Make new get request pai/profile/github/:username. Bring in ‘request’ and ‘config’.
* const options = {
* uri: `https://api.github.com/users/${req.params.username}/repos?per\_page=5&sort=created:asc&client\_id=${config.get('githubClientId')}&client\_secret=${config.get('githubSecret')}`,
* method: 'GET',
* headers: {'user-agent': 'node.js'}
* };

* **Posts** – create post model, models/Post.js
* Post routes. Routes/api/posts. Bring in express validator, middleware.
* Get and delete Post routes
* **Post like and unlike routes**
* **Add and remove comment routes**