1. App.js

**a)const bodyParser = require('body-parser');**

This code imports the 'body-parser' module, which is a **middleware** for handling HTTP requests in Node.js. It allows you to parse incoming request bodies in a middleware before your handlers, available under the req.body property. This is useful when receiving JSON or other structured data in the body of an HTTP request.

**b)const app = express();**

**app.set('view engine', 'ejs');**

**app.set('views', 'views');**

This code creates an instance of the Express.js framework and sets it up to use the EJS template engine for rendering views. The first line creates an instance of the Express.js application, which is commonly referred to as "app" in Express.js applications.

The second line sets the 'view engine' property to 'ejs', which tells Express.js to use the EJS template engine to render views.

The third line sets the 'views' property to 'views', which tells Express.js to look for views in a folder called 'views' in the root of the application. This is the default location for views in an Express.js application, but it can be configured to look in a different location if desired**.**

**c)const path = require('path');**

**module.exports = path.dirname(process.mainModule.filename);**

This code is using the path module in Node.js to get the directory path of the main module of the application.

The require('path') statement is used to import the built-in path module in Node.js.

The path.dirname() function is then used to get the directory path of the file passed as an argument. In this case, process.mainModule.filename is passed as the argument, which is a global variable in Node.js that contains the file path of the main module of the application.

Finally, the module.exports statement is used to export the directory path as a module that can be imported and used in other parts of the application.

This code is useful for getting the base directory of the project, which can be useful for referencing other files or directories within the project. For example, you could use this code to reference a specific directory within your project, like so:

/other/directory');

*const basePath = require('./path');*

*const filePath = path.join(basePath, 'some/other/directory');*

This code would join the base directory of the project with the specified directory, giving you the full path to that directory. This way you can have a more dynamic way to reference paths and files in your project.

**const errorController = require('./controllers/error');**

This code is requiring a module from a file called "error" located in a subdirectory called "controllers". The module is being assigned to a variable called "errorController". This variable can then be used to access the functionality exported by the module in the "error" file.

**const sequelize = require('./util/database');**

This code is requiring a module from a file called "database" located in a subdirectory called "util". The module is being assigned to a variable called "sequelize". The module is likely a configuration file that exports a Sequelize instance which can be used to interact with a database. The variable "sequelize" can then be used to connect to the database and perform operations like creating tables, inserting data, and querying data.

**var cors = require('cors');**

This code is requiring the "cors" module. CORS (Cross-Origin Resource Sharing) is a mechanism that allows a server to relax the same-origin policy, which is a security feature implemented by web browsers. It allows a web page to make requests to a different domain than the one it came from. The "cors" module is a Node.js package that provides middleware for Express.js and Connect.js, which can be used to enable CORS support in a web application. By requiring this package, it enables the web application to handle CORS related requests, allowing communication between different domains.

**app.use(bodyParser.json());**

This code is using the "body-parser" middleware in the Express.js app. "body-parser" is a Node.js package that helps to parse incoming request bodies in a middleware before your handlers, available under the req.body property. The "bodyParser.json()" method is used to parse the request body and make it available in the request object (req) as a JavaScript object. This allows the server to easily access the data sent in the request body, which is commonly used in API requests where data is sent in the request body, such as when sending JSON data.