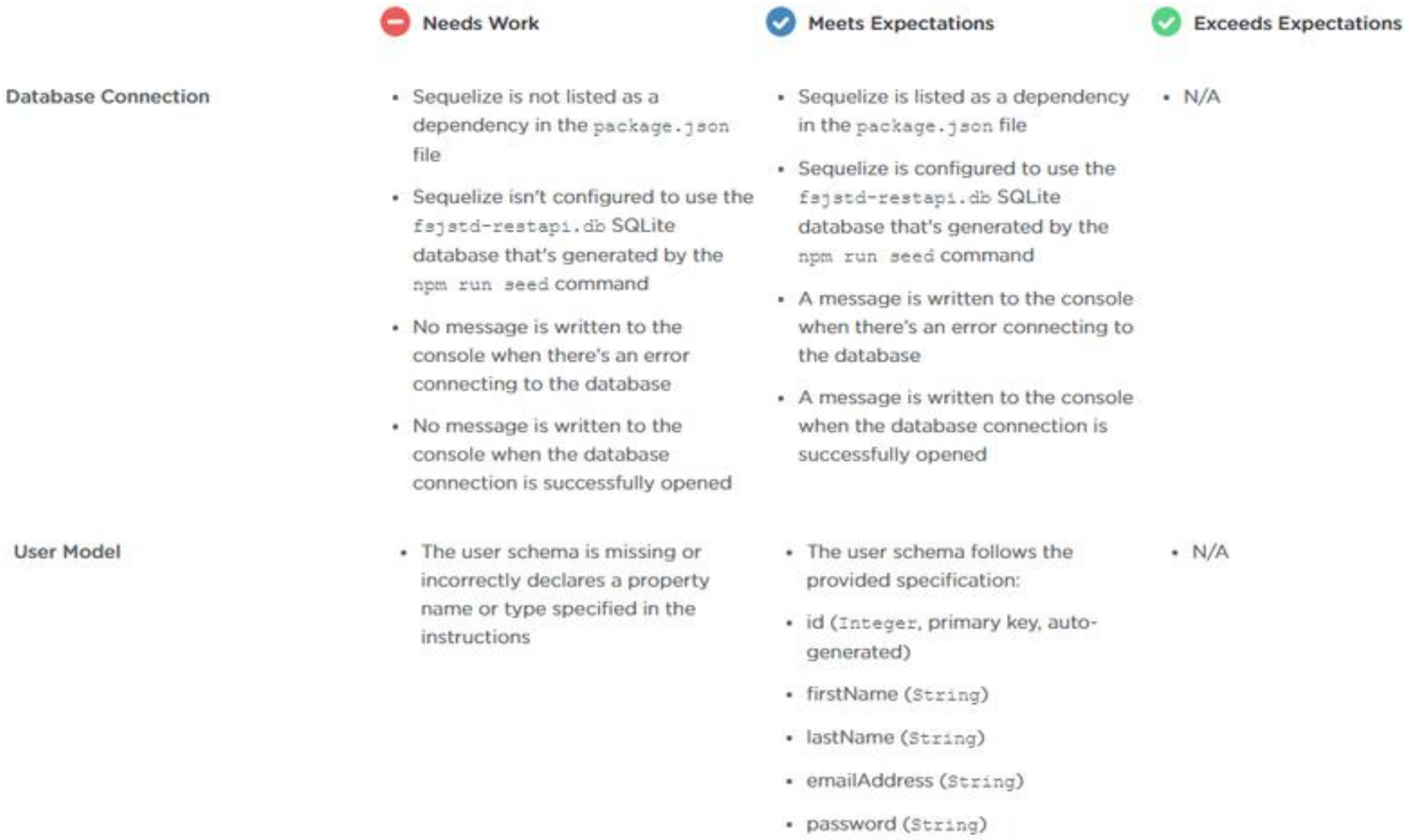
REST API CHECKLIST



	Needs Work	Meets Expectations	Exceeds Expectations
Database Connection	• Sequelize is not listed as a dependency in the `package.json` file • Sequelize isn't configured to use the `fsjstd-restapi.db` SQLite database that's generated by the `npm run seed` command • No message is written to the console when there's an error connecting to the database • No message is written to the console when the database connection is successfully opened	• Sequelize is listed as a dependency in the `package.json` file • Sequelize is configured to use the `fsjstd-restapi.db` SQLite database that's generated by the `npm run seed` command • A message is written to the console when there's an error connecting to the database • A message is written to the console when the database connection is successfully opened	• N/A
User Model	• The user schema is missing or incorrectly declares a property name or type specified in the instructions	• The user schema follows the provided specification: • id (`Integer`, primary key, auto-generated) • firstName (`String`) • lastName (`String`) • emailAddress (`String`) • password (`String`)	• N/A

Course Model	• The course schema is missing or incorrectly declares a property name or type specified in the instructions	• The course schema follows the provided specification: • id (`Integer`, primary key, auto-generated) • userId (`id` from the `Users` table) • title (`String`) • description (`Text`) • estimatedTime (`String`, nullable) • materialsNeeded (`String`, nullable)	• N/A
Model Associations	• A HasMany association isn't defined between the `User` and `Course` models • A BelongsTo association isn't defined between the `Course` and `User` models	• A HasMany association is defined between the `User` and `Course` models • A BelongsTo association is defined between the `Course` and `User` models	• N/A
User Routes	• One or more of the user routes specified are unavailable or are returning data not formatted as specified	• All of the following user routes are available and return the following data as specified: • `GET /api/users 200` - Returns the currently authenticated user • `POST /api/users 201` - Creates a user, sets the `Location` header to "/", and returns no content	• The `GET /api/users` route filters out the following properties: • `password` • `createdAt` • `updatedAt`

Course Routes	• One or more of the following course routes are unavailable or are returning data not formatted as specified	• All of the following course routes are available and return the following data as specified: • `GET /api/courses 200` - Returns a list of courses (including the user that owns each course) • `GET /api/courses/:id 200` - Returns a the course (including the user that owns the course) for the provided course ID • `POST /api/courses 201` - Creates a course, sets the `Location` header to the URI for the course, and returns no content • `PUT /api/courses/:id 204` - Updates a course and returns no content • `DELETE /api/courses/:id 204` - Deletes a course and returns no content	• The `GET /api/courses` and `/api/courses/:id` routes filter out the following properties: • `createdAt` • `updatedAt` • The `PUT /api/courses/:id` and `DELETE /api/courses/:id` routes return a `403` status code if the current user doesn't own the requested course

Validations	• The `POST` and `PUT` routes don't validate if the request body doesn't contain the following required values: • User ◦ firstName ◦ lastName ◦ emailAddress ◦ password • Course ◦ title ◦ description • Validation error(s) aren't sent with a`400` status code	• The `POST` and `PUT` routes validate if the request body contains the following required values: • User ◦ firstName ◦ lastName ◦ emailAddress ◦ password • Course ◦ title ◦ description • Validation error(s) are sent with a`400` status code	• The `POST /api/users` route validates that the provided email address is a valid email address and isn't already associated with an existing user
Password Security	• The `POST /api/users` route doesn't hash the user's password before persisting the user to the database	• The `POST /api/users` route hashes the user's password before persisting the user to the database	• N/A
Permissions	• No Express middleware function present to authenticate one or more of the specified routes • When authentication fails a `401` status code isn't returned	• An Express middleware function authenticates the following routes: • `GET /api/users` • `POST /api/courses` • `PUT /api/courses/:id` • `DELETE /api/courses/:id` • When authentication fails a `401` status code is returned	• N/A