```
// Below is a number of instructions. You need to write the appropriate
code to
// satisfy the requirements.
// DO NOT modify any code I have provided you with!

const express = require('express');
const sqlite3 = require('sqlite3').verbose();

/* INSTRUCTION 1
    Connect to the database file here, the connection should be stored
    in a variable called db
    If there is an error, this should be logged to the console
    If there is no error, a message confirming the database has been
connected to
    should be logged to the console.
*/
// Write code for instruction 1 here

const app = express();

/* INSTRUCTION 2
    You will need to write 4 lines of code here:
    1. Set the view engine for node to "ejs"
    2. Tell node it can find the static resources in the "public folder"
    3 & 4: Tell node to convert incoming requests to JSON
        - you'll need to do this to be able to handle POST requests
        - There is 2 lines of code for this part - see lectures
*/
// Write code for instruction 2 here

/* INSTRUCTION 3
    Tell the server to start listening for requests.
    It must listen on port 5000!
    A message stating that the server has started should be logged to the
console
*/
// Write code for instruction 3 here


app.get("/api/team", (req, res)=>{
    /* INSTRUCTION 4
    Query the database - get all teams.
    This route should provide the results of the query in json format
    */
    // Write code for instruction 4 here
});

app.get("/api/team/:id", (req, res)=>{
    /* INSTRUCTION 5
    Query the database - using the parameter, get the information of a
single
    team from the database.
    This route should respond with data in json format
    */
    // Write code for instruction 5 here
});

app.post("/api/team", (req, res)=>{
    /* INSTRUCTION 6
    This route should receive information in JSON format.
    It should then add this data to the database
```

```
    The responce should be in json format and should just indicate whether
or not the
    record was successfully added to the database
    */
    // Write code for instruction 6 here
});

app.delete("/api/team/:id", (req, res)=>{
    /* INSTRUCTION 7
    This route receives a parameter that represents a team ID.
    It should delete this team from the database
    The responce should be in json format and should just indicate whether
or not the
    record was successfully deleted
    */
    // Write code for instruction 7 here
});


app.put("/api/team/:id", (req, res)=>{
    /* INSTRUCTION 8 (C grade)
    NOTE. this is a C grade task. You should prioritise the tasks below
first.
    This route should receive information in JSON format and a parameter
representing
    team ID.
    It should then update the details of the appropriate team
    The responce should be in json format and should just indicate whether
or not the
    record was successfully updated. You may also return the new
    details of the updated record as part of the json response
    */
    // Write code for instruction 8 here
});


/* INSTRUCTIONS
    For the remainder of the document, you will need to use the routes
below, as required in
    the 'website' section of the assessment brief.
    You should use EJS to render HTML for the remianing routes.
    For a low pass, the website shoud allow
        - Viewing
        - Adding
        - Deleting
    You will need to create additional routes to do this
    You do not need to connect to the API routes to achieve this
    Only when attempting A grade tasks does the API need to be communicated
with.

    More requirements are added in the grading criteria (see assessment
brief)
*/

app.get("/", (req, res)=>{
    /* This route should render your home page
    */
});

app.get("/team", (req, res)=>{
```

```
    /* This route should render a webpage that lists all teams from the
database
    */
});

app.get("/team/:id", (req, res)=>{
    /* This route should display information about a specific team (based
on the
        id parameter)
    */
});

/*
    You will now need to create your own routes that facilitate the adding
and
    deleting of teams. This is a website, so everything should be displayed
to the user
    as a webpage.

    For a C grade you should also allow the user to update team
information.

    See grading criteria for further C, B and A grade tasks. No prewritten
code is
    provided for these tasks.
*/
```