// Importing the express module to create the server and routes import express from 'express';

// Importing mysql2 module to interact with MySQL database import mysql from 'mysql2';

// Creating an instance of the express application const app = express();

// Defining the port number where the server will listen const PORT = 3000;

// -- // MIDDLEWARE // --

// Use express.json() to parse incoming JSON request bodies // This allows the server to understand JSON data sent in POST, PUT requests app.use(express.json());

// -- // MYSQL DATABASE CONNECTION // --

// Create a connection object to connect with MySQL database const db = mysql.createConnection({ host: 'localhost', // Host where the database is running user: 'root', // MySQL username password: 'cdac', // MySQL password database: 'cdac_wpt', // Name of the database you want to connect to });

// Connecting to the MySQL database db.connect(err => { if (err) { // If there's an error while connecting, throw the error and stop the server throw err; } // If connected successfully, log a success message console.log("MySQL connected successfully!!"); });

// -- // ROUTES // --

// 1. CREATE: Add a new book // This route handles HTTP POST requests to '/book' app.post('/book', (req, res) => { // Extract bookid, bookname, and price from the request body const { bookid, bookname, price } = req.body;

// Validate the input - all fields must be present if (!bookid || !bookname || !price) { return res.status(400).json({ error: 'All fields are required' }); }

// SQL query to insert the book data into the book table const sql = 'INSERT INTO book (bookid, bookname, price) VALUES (?, ?, ?)';

// Execute the SQL query db.query(sql, [bookid, bookname, price], (err) => { if (err) { // If there's a DB error, return a 500 status with error message return res.status(500).json({ error: err.message }); }

```
// On success, send a JSON response
res.json({ message: 'Book Created Successfully!' });
```

}); });

// 2. READ: Get all books // This route handles HTTP GET requests to '/book' app.get('/book', (req, res) => { // SQL query to fetch all records from the book table const sql = 'SELECT * FROM book';

// Execute the SQL query db.query(sql, (err, results) => { if (err) { // If error, send 500 status with error message return res.status(500).json({ error: err.message }); }

```
// On success, send all records as JSON
res.json(results);
```

}); });

// 3. READ: Get a specific book by ID // This route handles HTTP GET requests to '/book/:id' app.get('/book/:id', (req, res) => { // SQL query to fetch a specific book using bookid const sql = 'SELECT * FROM book WHERE bookid = ?';

// Execute the SQL query with the id from the URL parameter db.query(sql, [req.params.id], (err, results) => { if (err) { // If error, send 500 status return res.status(500).json({ error: err.message }); }

```
// If book found, send it; otherwise send an error message
res.json(results[0] || { error: 'Book not found' });
```

}); });

// 4. UPDATE: Update a book's name and price // This route handles HTTP PUT requests to '/book/:id' app.put('/book/:id', (req, res) => { // Extract updated fields from request body const { bookname, price } = req.body;

// SQL query to update bookname and price for a specific bookid const sql = 'UPDATE book SET bookname = ?, price = ? WHERE bookid = ?';

// Execute the update query db.query(sql, [bookname, price, req.params.id], (err) => { if (err) { // If error, send 500 status return res.status(500).json({ error: err.message }); }

```
// On success, send confirmation
res.json({ message: 'Book updated successfully' });
```

}); });

// 5. DELETE: Delete a book by ID // This route handles HTTP DELETE requests to '/book/:id' app.delete('/book/:id', (req, res) => { // SQL query to delete a book based on bookid const sql = 'DELETE FROM book WHERE bookid = ?';

// Execute the delete query db.query(sql, [req.params.id], (err) => { if (err) { // If error, send 500 status return res.status(500).json({ error: err.message }); }

```
// On success, send confirmation
res.json({ message: 'Book deleted successfully' });
```

}); });

// -- // START SERVER // ----------------------------

// Start the server and listen on the defined PORT app.listen(PORT, () => { console.log(`Server running at http://localhost:${PORT}`); });