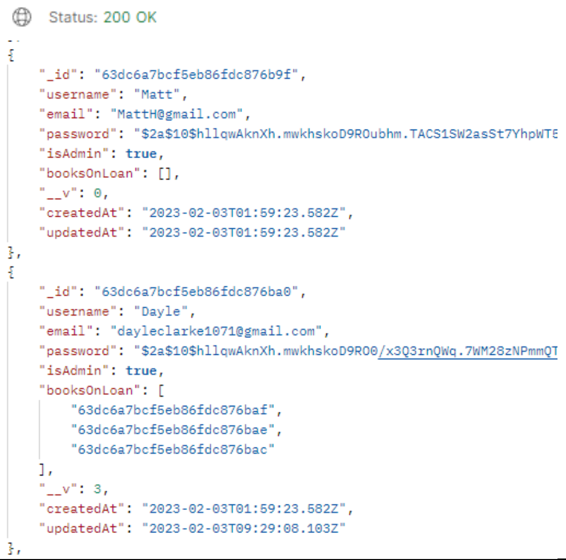
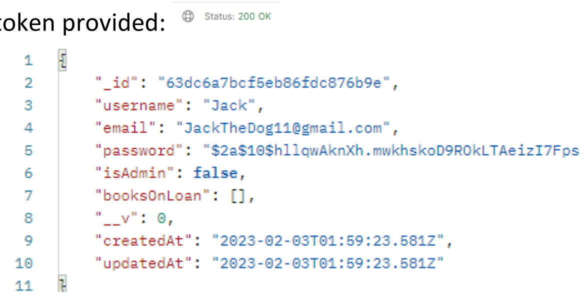
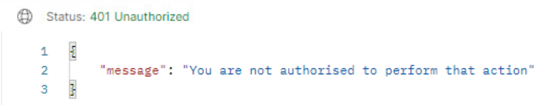
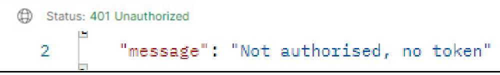
Backend Development API Testing in PostMan

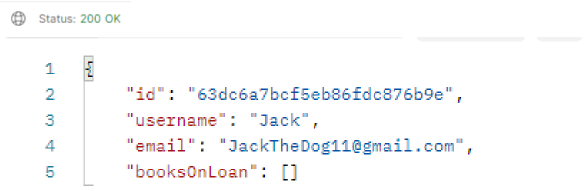
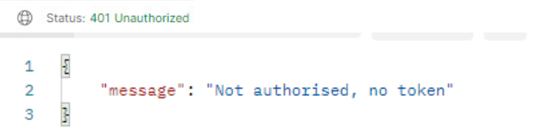
User-Testing of Back-End Development Site

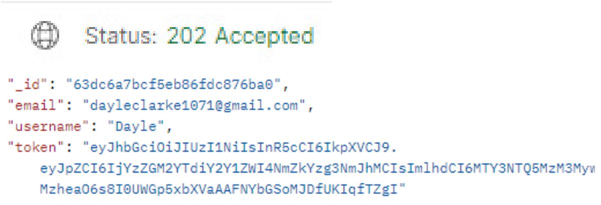
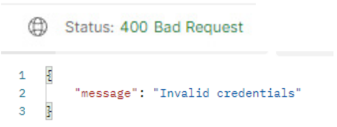
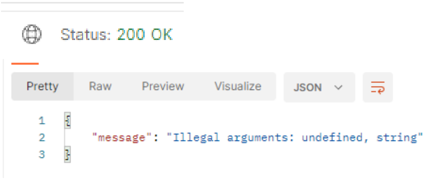
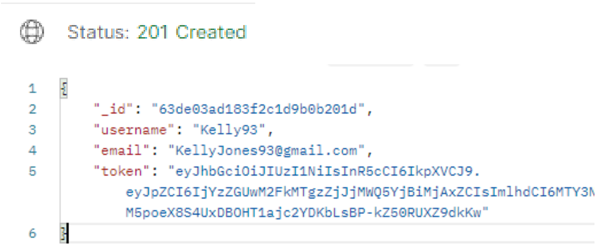
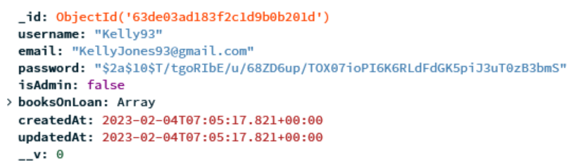
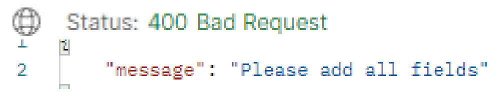
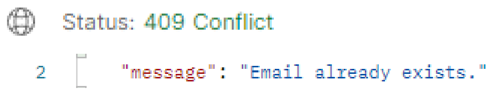
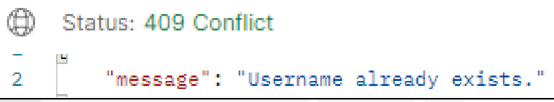
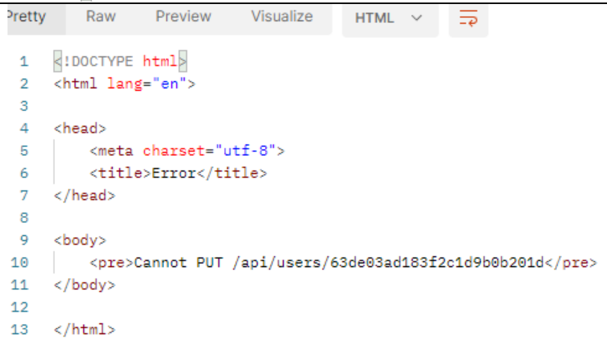
The following manual testing was run using Postman to test to endpoints of the API during development:

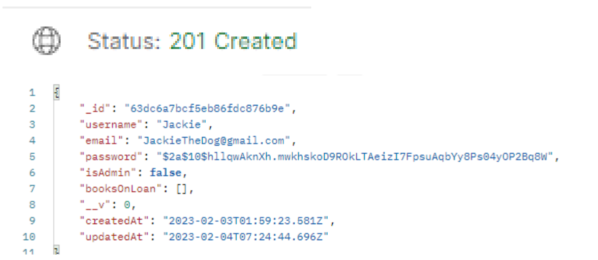
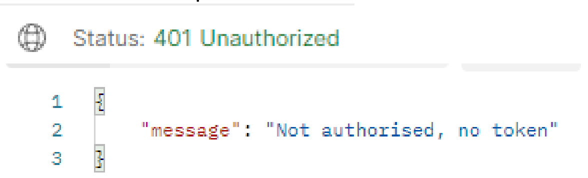
Routes Relating to Users (FULL CRUD functionality was provided and tested on the back end in development and after deployment.			
End Point	**Expected Result**	**Actual Result**	**Issues Encountered/ Actions to Take**

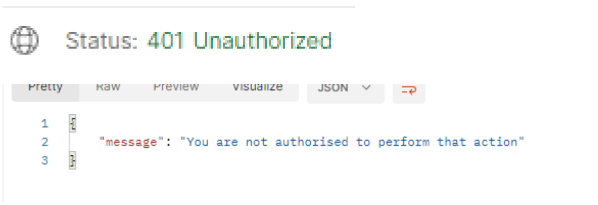
http://localhost:5000/api/users/	**GET All Users**
	Should display a list of all users with the following fields:
	• Id (autogenerated, 24 characters))
	• Username
	• Email
	• Password (hashed)
	• IsAdmin
	• The date the user was created and last updated
	• A nested array containing all the books the user currently has on loan.
	Should return a 200 status code
		The following is displayed regardless of whether an authentication token is provided:
		
		Status: 200 OK
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876b9f",`
		`"username": "Matt",`
		`"email": "MattH@gmail.com",`
		`"password": "$2a$10$hllqwAknXh.mwkhskoD9ROubhm.TACS1SW2asSt7YhpWT5`
		`"isAdmin": true,`
		`"booksOnLoan": [],`
		`"__v": 0,`
		`"createdAt": "2023-02-03T01:59:23.582Z",`
		`"updatedAt": "2023-02-03T01:59:23.582Z"`
		`},`
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876ba0",`
		`"username": "Dayle",`
		`"email": "dayleclarke1071@gmail.com",`
		`"password": "$2a$10$hllqwAknXh.mwkhskoD9RO0/x3Q3rnQWq.7WM28zNPmmQT`
		`"isAdmin": true,`
		`"booksOnLoan": [`
		`"63dc6a7bcf5eb86fdc876baf",`
		`"63dc6a7bcf5eb86fdc876bae",`
		`"63dc6a7bcf5eb86fdc876bac"`
		`],`
		`"__v": 3,`
		`"createdAt": "2023-02-03T01:59:23.582Z",`
		`"updatedAt": "2023-02-03T09:29:08.103Z"`
		`},`
			The route is displaying regardless of whether an authentication token is provided. This is likely because the protection middleware has been disabled to debug issues during development.
			Even though the password is hashed this should be removed.
			
			To Do
			• Add back on the protect middleware so that this route requires an admin JWT token.
			• Remove the password from the output

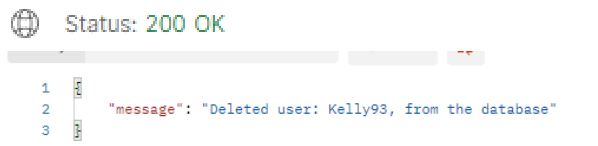
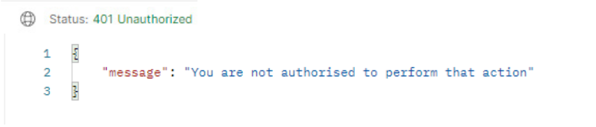
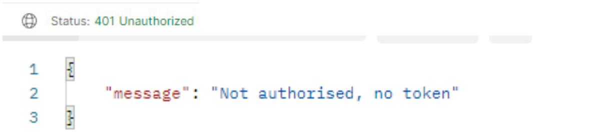
http://localhost:5000/api/users/:ID	**GET One User Based on their ID**
	• Should display information about the user with the ID provided in the URI with same fields as above. Should have a 200 status code
	• An error message should be displayed if the user attempting to access the route is not an admin user
	• An error message should be displayed if a token is not provided.
		Route with admin (used with isAdmin set to true) token provided:
		
		Status: 200 OK
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876b9e",`
		`"username": "Jack",`
		`"email": "JackTheDog11@gmail.com",`
		`"password": "$2a$10$hllqwAknXh.mwkhskoD9ROkLTAeizI7Fps`
		`"isAdmin": false,`
		`"booksOnLoan": [],`
		`"__v": 0,`
		`"createdAt": "2023-02-03T01:59:23.581Z",`
		`"updatedAt": "2023-02-03T01:59:23.581Z"`
		`}`
		Error Returned when a token is provided but the user has isAdmin set to false.
		
		Status: 401 Unauthorized
		`{`
		`"message": "You are not authorised to perform that action"`
		`}`
		Error Returned when no token is provided:
		
		Status: 401 Unauthorized
		`"message": "Not authorised, no token"`
			Password is displayed in the response.
			
			To-Do:
			• Remove password from this route.

http://localhost:5000/api/users/profile	**GET User's Own Profile Based on the Token Provided**
	• Should display the information as above for the user who's token was provided. Should have a 200 status code
	• An error message should be displayed if a token is not provided.
		Endpoint returned when Jack's token is provided.
		
		Status: 200 OK
		`{`
		`"id": "63dc6a7bcf5eb86fdc876b9e",`
		`"username": "Jack",`
		`"email": "JackTheDog11@gmail.com",`
		`"booksOnLoan": []`
		Error Returned when no token is provided:
		
		Status: 401 Unauthorized
		`{`
		`"message": "Not authorised, no token"`
		`}`
			A number of fields here are missing. Depending on what this route is used for on the back end this may need to change.

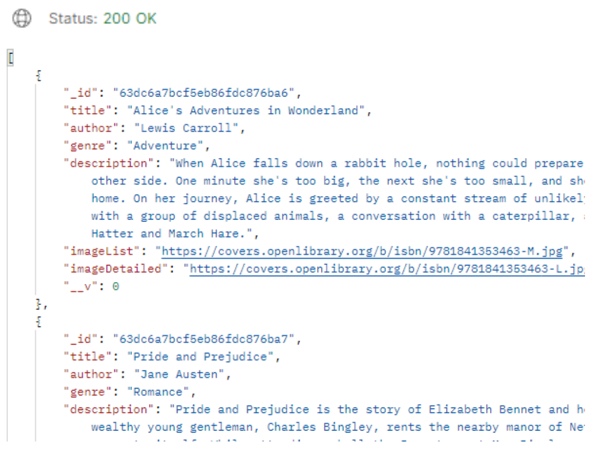
http://localhost:5000/api/users/login	**POST- Login a User** When a correct username and password are passed in as a JSON body it should return the user's credentials and a JWT token. Should have a 202 status code. When incorrect details are provided an error message should be displayed in JSON indicating that their credentials are incorrect	When correct details are provided:  Status: 202 Accepted `"_id": "63dc6a7bcf5eb86fdc876ba0",` `"email": "dayleclarke1071@gmail.com",` `"username": "Dayle",` `"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzZGM2YTdiY2Y1ZWI4NmZkYzg3NmJhMCIsImlhdCI6MTY3NTQ5MzM3Myw Mzhea06s8I0UWGp5xbXVaAAFNYbGSoMJDfUKIqfTZgI"` When incorrect details are passed in:  Status: 400 Bad Request `{` `"message": "Invalid credentials"` `}` When either the username or password is missing:  Status: 200 OK `{` `"message": "Illegal arguments: undefined, string"` `}`	400 bad request is not specific enough for incorrect credentials. This should be 401 unauthorised. The message returned when a field is missing is not specific and it doesn't return an error status code to inform the user that they are missing a required field. **To-Do:** • Change Status code to 401 Bad request • Add in error handling to handle the situation where the user omits a required field.
http://localhost:5000/api/users/register	**POST- Register a User** When a unique username and email are provided along with a password a new user is created and added to the users collection in the database. An error message with a 409 conflict error should be provided if a username or email has already been taken and the message should specifically state which one is missing. An error message should be displayed if a username, email or password is missing.	When all the correct fields are provided:  Status: 201 Created `{` `"_id": "63de03ad183f2c1d9b0b201d",` `"username": "Kelly93",` `"email": "KellyJones93@gmail.com",` `"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzZGUwM2FkMTgzZjJjMWQ5YjBiMjAxZCIsImlhdCI6MTY3N M5poeX8S4UxDBOHT1ajc2YDKbLsBP-kZ50RUXZ9dkKw"` `}` The password has also been added to the database and hashed. This is shown on MongoDB Atlas:  `_id: ObjectId('63de03ad183f2c1d9b0b201d')` `username: "Kelly93"` `email: "KellyJones93@gmail.com"` `password: "$2a$10$T/tgoRIbE/u/68ZD6up/TOX07ioPI6K6RLdFdGK5piJ3uT0zB3bmS"` `isAdmin: false` `> booksOnLoan: Array` `createdAt: 2023-02-04T07:05:17.821+00:00` `updatedAt: 2023-02-04T07:05:17.821+00:00` `__v: 0` When a field is missing:  Status: 400 Bad Request `"message": "Please add all fields"` When the email is already taken:  Status: 409 Conflict `"message": "Email already exists."` When the username is already taken:  Status: 409 Conflict `"message": "Username already exists."`	Could add a more specific message for when a field is missing in the JSON body. Research whether there is a more specific error code for a missing field.
http://localhost:5000/api/users/:ID	**Update a User Based on Their ID** These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.	 Pretty Raw Preview Visualize HTML `<!DOCTYPE html>` `<html lang="en">` `<head>` `<meta charset="utf-8">` `<title>Error</title>` `</head>` `<body>` `<pre>Cannot PUT /api/users/63de03ad183f2c1d9b0b201d</pre>` `</body>` `</html>`	The update route in the userRoutes was not actually provided. Decide whether this route is required. Will admin need to update information on other users? Is this MVP? Add error handling so that this error is presented to the user in JSON

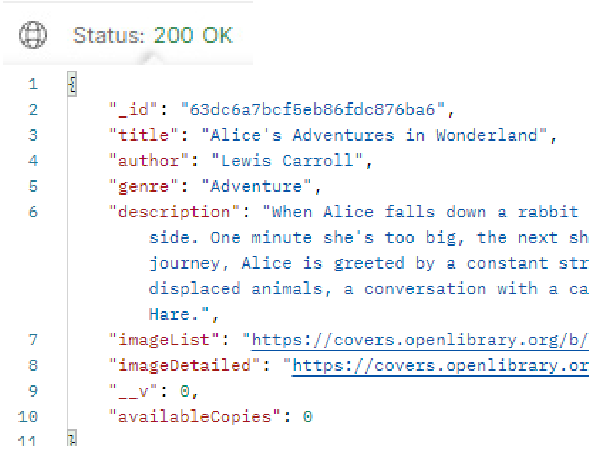
http://localhost:5000/api/users/profile	**PUT: Update a user based on the Token passed in.**
	Should allow one or more fields to be updated based on the JSON body provided in the request body. If no token id provided a 401 error should be raised and a not authorised, no token message provided.
		Response when one or more fields are updated (the email and username successfully changed). This was tested with one and multiple fields provided:
		
		Status: 201 Created
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876b9e",`
		`"username": "Jackie",`
		`"email": "JackieTheDog@gmail.com",`
		`"password": "$2a$10$hllqwAknXh.mwkhskoD9ROkLTAeizI7FpsuAqbYy8Ps04yOP2Bq8W",`
		`"isAdmin": false,`
		`"booksOnLoan": [],`
		`"__v": 0,`
		`"createdAt": "2023-02-03T01:59:23.581Z",`
		`"updatedAt": "2023-02-04T07:24:44.696Z"`
		`}`
		When no token is provided:
		
		Status: 401 Unauthorized
		`{`
		`"message": "Not authorised, no token"`
		`}`
			Remove the password from the output.

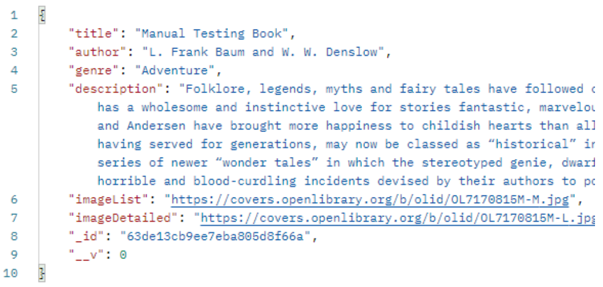
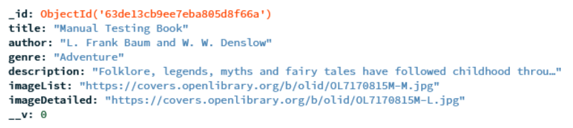
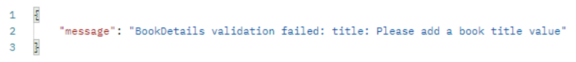
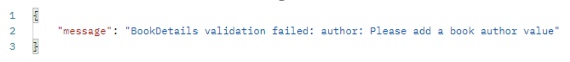
http://localhost:5000/api/ users/profile	**DELETE User based on the provided token.**
	The user who owns the token provided in the authorisation header should be deleted from the system.
		When the user's token is provided a delete request returns the following error message:
		
		Status: 401 Unauthorized
		`{`
		`"message": "You are not authorised to perform that action"`
		`}`
			This route isn't allowing the owner of the token to delete their own profile. To-Do: Adjust the authorisation on this to allow a user to delete their own profile. If the token belongs to the user they should be able to delete their account.

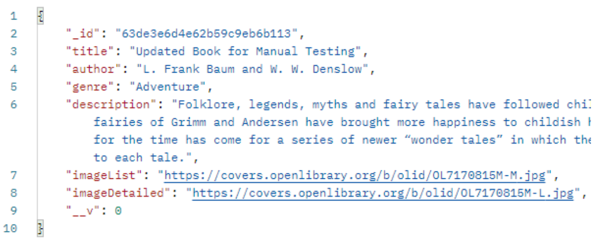
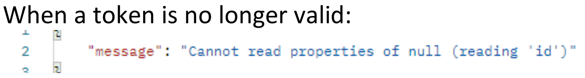
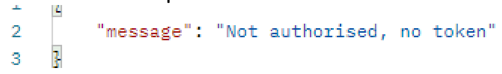
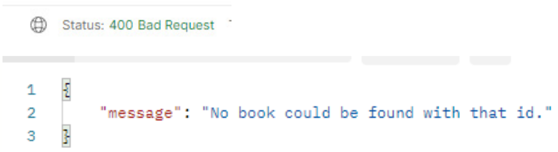
http://localhost:5000/api/users/63de03ad183f2c1d9b0b20 1d	**DELETE User Based on the Id provided as a URI parameter.**
	• When a user's id is passed in as a parameter to the uri that user should be deleted from the DB. A message confirming this deletion and a 200 error should be returned.
	• An error message should be displayed if the user attempting to access the delete route is not an admin user
	• An error message should be displayed if a token is not provided.
		Message returned when an admin token is provided:
		
		Status: 200 OK
		`{`
		`"message": "Deleted user: Kelly93, from the database"`
		`}`
		After checking the MongoDB ATLAS and also the get All users route it was confirmed that Kelly93 was removed from the DB
		Error Returned when a token is provided but the user has isAdmin set to false.
		
		Status: 401 Unauthorized
		`{`
		`"message": "You are not authorised to perform that action"`
		`}`
		Error Returned when no token is provided:
		
		Status: 401 Unauthorized
		`{`
		`"message": "Not authorised, no token"`
		`}`	

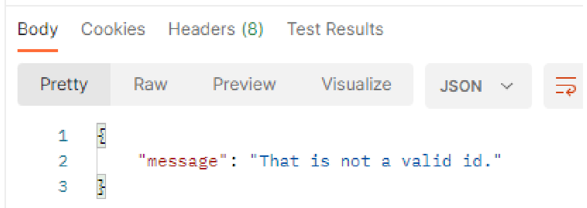
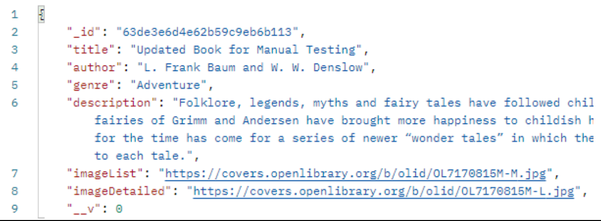
Routes Relating to Book Details (FULL CRUD functionality was provided an tested on the back end in development and after deployment.			
End Point	**Expected Result**	**Actual Result**	**Issues Encountered/ Actions to Take**

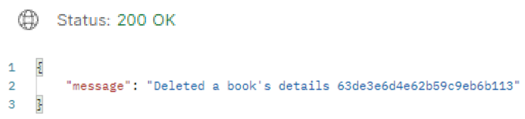
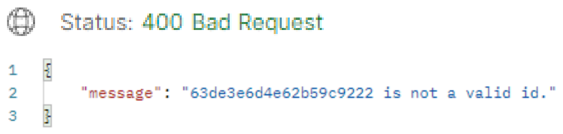
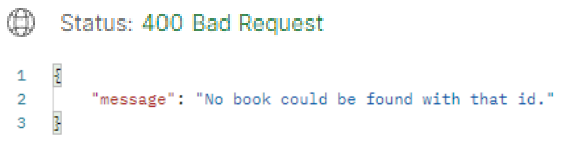
http://localhost:5000/api/bookdetails/	**GET All BookDetails (no token required)**
	Should display a list of all book details in an array with the following fields:
	• Id (autogenerated, 24 characters))
	• Title
	• Author
	• Genre
	• Description
	• imageList
	• imagedetails
	Should have a 200 status code
		
		Status: 200 OK
		`[`
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876ba6",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit hole, nothing could prepare`
		`other side. One minute she's too big, the next she's too small, and sh`
		`home. On her journey, Alice is greeted by a constant stream of unlikel`
		`with a group of displaced animals, a conversation with a caterpillar,`
		`Hatter and March Hare.",`
		`"imageList": "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg",`
		`"imageDetailed": "https://covers.openlibrary.org/b/isbn/9781841353463-L.jp`
		`"__v": 0`
		`},`
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876ba7",`
		`"title": "Pride and Prejudice",`
		`"author": "Jane Austen",`
		`"genre": "Romance",`
		`"description": "Pride and Prejudice is the story of Elizabeth Bennet and h`
		`wealthy young gentleman, Charles Bingley, rents the nearby manor of Ne`
			No issues raised. Consider adding a method to determine how many copies of the book are available.
			
			Images are stored as URL's. Research how these should be stored as best practice.

http://localhost:5000/api/bookDetails/:id	**GET One BookDetails** (no token required)
	Should display a single book with all of its details (as above) based on the Book Details ID provided
		
		Status: 200 OK
		`{`
		`"_id": "63dc6a7bcf5eb86fdc876ba6",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit`
		`side. One minute she's too big, the next sh`
		`journey, Alice is greeted by a constant str`
		`displaced animals, a conversation with a ca`
		`Hare.",`
		`"imageList": "https://covers.openlibrary.org/b/`
		`"imageDetailed": "https://covers.openlibrary.or`
		`"__v": 0,`
		`"availableCopies": 0`
		`}`
			Images are stored as URL's. Research how these should be stored as best practice.

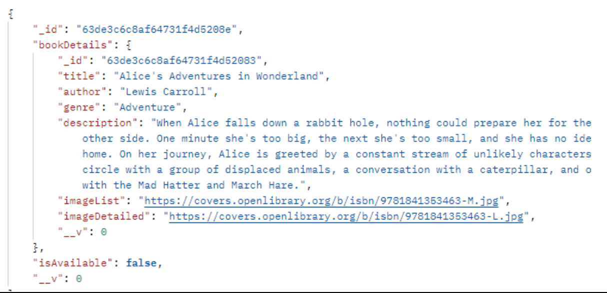
http://localhost:5000/api/bookDetails/	**Post new Book Details**
	• A new entry in the BookDetails collection in the DB should be created
	• An error message should be displayed if the user attempting to access the route is not an admin user
	• An error message should be displayed if a token is not provided.
		Result returned when posting new book details regardless of whether a token is provided.
		
		`{`
		`"title": "Manual Testing Book",`
		`"author": "L. Frank Baum and W. W. Denslow",`
		`"genre": "Adventure",`
		`"description": "Folklore, legends, myths and fairy tales have followed c`
		`has a wholesome and instinctive love for stories fantastic, marvelou`
		`and Andersen have brought more happiness to childish hearts than all`
		`having served for generations, may now be classed as "historical" ir`
		`series of newer "wonder tales" in which the stereotyped genie, dwarf`
		`horrible and blood-curdling incidents devised by their authors to pc`
		`"imageList": "https://covers.openlibrary.org/b/olid/OL7170815M-M.jpg",`
		`"imageDetailed": "https://covers.openlibrary.org/b/olid/OL7170815M-L.jpg`
		`"_id": "63de13cb9ee7eba805d8f66a",`
		`"__v": 0`
		`}`
		This new entry now appears in the list of bookdetails when the get all route is used in Postman and also in MongoDB on Atlas as shown here:
		
		`_id: ObjectId('63de13cb9ee7eba805d8f66a')`
		`title: "Manual Testing Book"`
		`author: "L. Frank Baum and W. W. Denslow"`
		`genre: "Adventure"`
		`description: "Folklore, legends, myths and fairy tales have followed childhood throu…"`
		`imageList: "https://covers.openlibrary.org/b/olid/OL7170815M-M.jpg"`
		`imageDetailed: "https://covers.openlibrary.org/b/olid/OL7170815M-L.jpg"`
		`__v: 0`
		When a title is missing:
		
		`{`
		`"message": "BookDetails validation failed: title: Please add a book title value"`
		`}`
		When an author is missing:
		
		`{`
		`"message": "BookDetails validation failed: author: Please add a book author value"`
		`}`
			This route doesn't return the ID which might be required depending on the front end route.
			The new book details route is posting regardless of whether an authentication token is provided or not. This is because the protection middleware has been disabled to debug issues during development.
			To Do
			Add pack on the protect middleware so that this route requires an admin JWT token

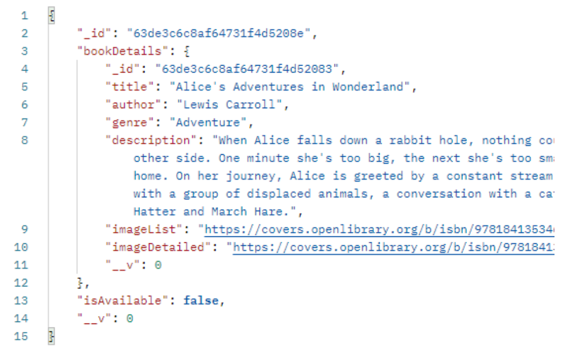
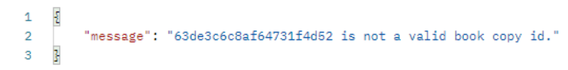
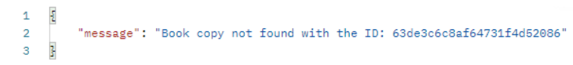
http://localhost:5000/api/bookDetails/:id	**Update (PUT) One BookDetails Entry**
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		Put request when one or more fields are provided:
		`{`
		`"_id": "63de3e6d4e62b59c9eb6b113",`
		`"title": "Updated Book for Manual Testing",`
		`"author": "L. Frank Baum and W. W. Denslow",`
		`"genre": "Adventure",`
		`"description": "Folklore, legends, myths and fairy tales have followed chi`
		`fairies of Grimm and Andersen have brought more happiness to childish h`
		`for the time has come for a series of newer “wonder tales” in which the`
		`to each tale.",`
		`"imageList": "https://covers.openlibrary.org/b/olid/OL7170815M-M.jpg",`
		`"imageDetailed": "https://covers.openlibrary.org/b/olid/OL7170815M-L.jpg",`
		`"__v": 0`
		`}`
		When a token is no longer valid:
		`"message": "Cannot read properties of null (reading 'id')"`
		When no token is provided:
		`{`
		`"message": "Not authorised, no token"`
		`}`
		Error message thrown when a URI is passed in that isn’t 24 character long:
		"message": "Cast to ObjectId failed for value \"63de3e6d4e62b59c9eb6b14\" (type string) at path \"_id\" for model \"BookDetails\""
		Error message thrown when an id provided but it is invalid:
		Status: 400 Bad Request
		`{`
		`"message": "No book could be found with that id."`
		`}`
			The error message provided when a token is no longer valid is not descriptive. I didn’t understand why this error was occurring at first.
			
			The error that occurs when an id is passed in that isn’t in the correct format is not descriptive.

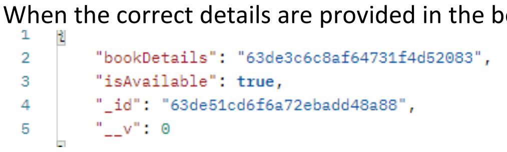
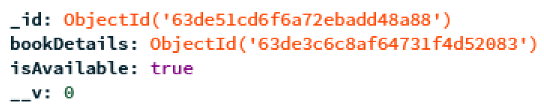
As before	**Update (PUT) One BookDetails Entry**
	• Change made to previous route to catch an invalid id.
	• Should return a more descriptive error message to inform the user that the id is not valid.
	• A valid id should still function as before.
		When an id less that is not 24 character is provided:
		Body Cookies Headers (8) Test Results
		Pretty Raw Preview Visualize JSON
		`{`
		`"message": "That is not a valid id."`
		`}`
		When a valid id is provided:
		`{`
		`"_id": "63de3e6d4e62b59c9eb6b113",`
		`"title": "Updated Book for Manual Testing",`
		`"author": "L. Frank Baum and W. W. Denslow",`
		`"genre": "Adventure",`
		`"description": "Folklore, legends, myths and fairy tales have followed chil`
		`fairies of Grimm and Andersen have brought more happiness to childish h`
		`for the time has come for a series of newer “wonder tales” in which the`
		`to each tale.",`
		`"imageList": "https://covers.openlibrary.org/b/olid/OL7170815M-M.jpg",`
		`"imageDetailed": "https://covers.openlibrary.org/b/olid/OL7170815M-L.jpg",`
		`"__v": 0`
			Could be most descriptive in the error message by restating the id provided.

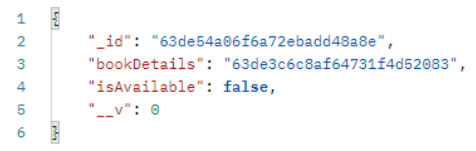
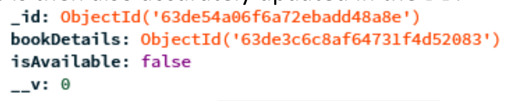
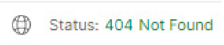
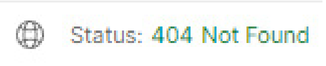
http://localhost:5000/api/bookDetails/:id	**DELETE one BookDetails based on the id passed in as URI parameters.**
	• This should provide a message in JSON indicating that the delete was successful and the entry should be removed from the db.
	• Error messages with an error status code should be provided is the id provided is not valid.
		When the id is valid:
		Status: 200 OK
		`{`
		`"message": "Deleted a book's details 63de3e6d4e62b59c9eb6b113"`
		`}`
		The books details is also deleted from the MongoDB on Atlas which was manually checked online.
		When an invalid ID is provided:
		Status: 400 Bad Request
		`{`
		`"message": "63de3e6d4e62b59c9222 is not a valid id."`
		`}`
		When the id provided is in the correct format but doesn’t belong to an entry in the collection:
		Status: 400 Bad Request
		`{`
		`"message": "No book could be found with that id."`
		`}`
			Perhaps change the message to read “Deleted the book details for the book with the id: …..”

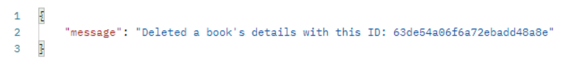
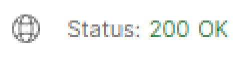
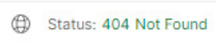
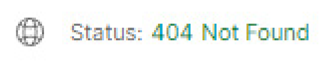
Routes Relating to Book Copies (FULL CRUD functionality was provided and tested			
End Point	**Expected Result**	**Actual Result**	**Issues Encountered/ Actions to Take**

http://localhost:5000/api/bookcopies	**Get All Book Copies**
	• Should return the book copies id, a Boolean indicating whether the book copy is available and a nested bookDetails object with details about the book.
		{
		"_id": "63de3c6c8af64731f4d5208e",
		"bookDetails": {
		"_id": "63de3c6c8af64731f4d52083",
		"title": "Alice's Adventures in Wonderland",
		"author": "Lewis Carroll",
		"genre": "Adventure",
		"description": "When Alice falls down a rabbit hole, nothing could prepare her for the other side. One minute she's too big, the next she's too small, and she has no ide home. On her journey, Alice is greeted by a constant stream of unlikely characters circle with a group of displaced animals, a conversation with a caterpillar, and o with the Mad Hatter and March Hare.",
		"imageList": "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg",
		"imageDetailed": "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg",
		"__v": 0
		},
		"isAvailable": false,
		"__v": 0
			N/A

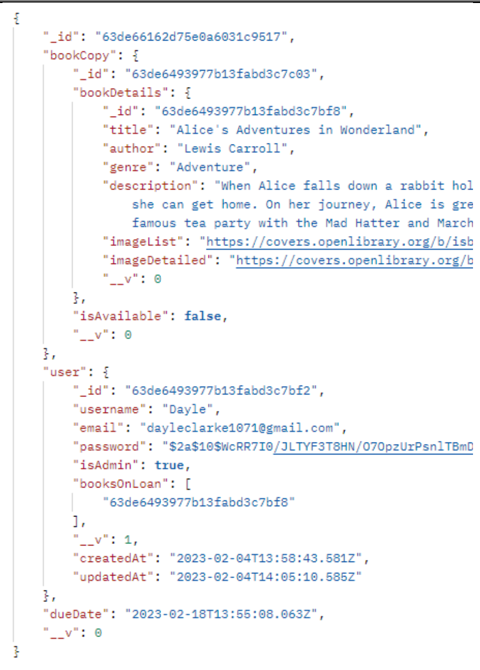
http://localhost:5000/api/bookcopies/:id	**GET a single book copy based on their id.**
	• Provide error messages if the id provided is not valid.
		When a correct id is provided:
		{
		"_id": "63de3c6c8af64731f4d5208e",
		"bookDetails": {
		"_id": "63de3c6c8af64731f4d52083",
		"title": "Alice's Adventures in Wonderland",
		"author": "Lewis Carroll",
		"genre": "Adventure",
		"description": "When Alice falls down a rabbit hole, nothing co other side. One minute she's too big, the next she's too sm home. On her journey, Alice is greeted by a constant stream with a group of displaced animals, a conversation with a ca Hatter and March Hare.",
		"imageList": "https://covers.openlibrary.org/b/isbn/97818413534
		"imageDetailed": "https://covers.openlibrary.org/b/isbn/9781841
		"__v": 0
		},
		"isAvailable": false,
		"__v": 0
		}
		Status: 404 Not Found
		When an invalid ID is provided:
		{
		"message": "63de3c6c8af64731f4d52 is not a valid book copy id."
		}
		When an ID is the correct format but doesn't belong
		Status: 404 Not Found
		to a book copy in the db:
		{
		"message": "Book copy not found with the ID: 63de3c6c8af64731f4d52086"
		}
			Should be consist with the error messages returned when the id is not valid. To-do research whether to use 400 or 404 and apply this consistently across the application

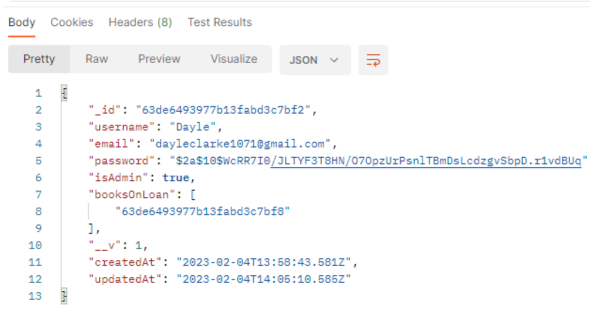
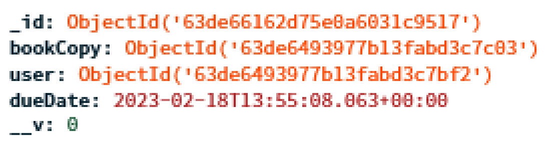
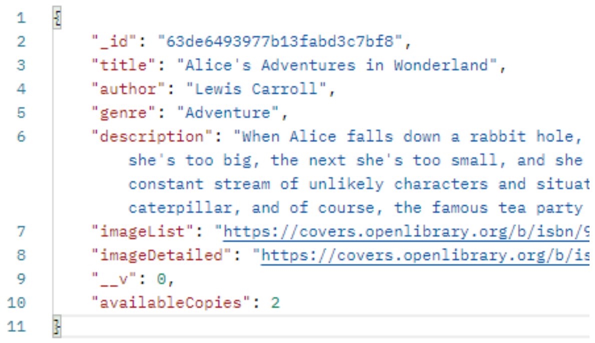
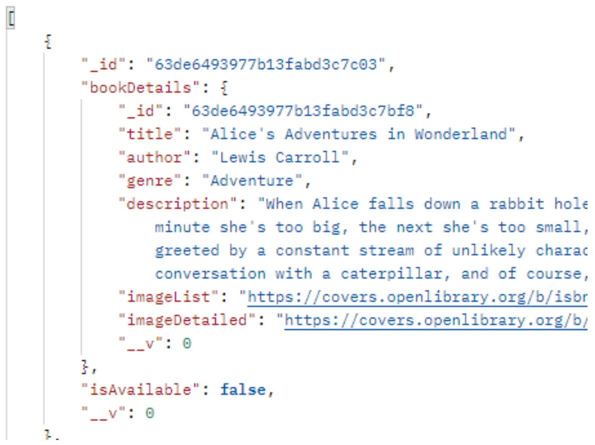
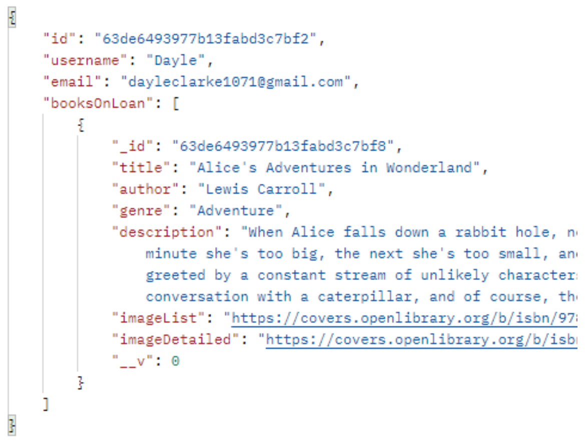
http://localhost:5000/api/bookcopies	**POST new book copy**
	• When the post request is made this should add the new book copy to the database.
	• Errors should be thrown if a valid bookDetails id is not provided and the isAvailable field should be added automatically.
		When the correct details are provided in the body:
		{
		"bookDetails": "63de3c6c8af64731f4d52083",
		"isAvailable": true,
		"_id": "63de51cd6f6a72ebadd48a88",
		"__v": 0
		}
		The isAvailable field is behaving as expected. It is accurately being added as true if the attribute is not provided in the body. If it is set to false in the body, it's also reflected as false in the DB.
		This also appears correctly in the MongoDB Atlas DB online:
		_id: ObjectId('63de51cd6f6a72ebadd48a88')
		bookDetails: ObjectId('63de3c6c8af64731f4d52083')
		isAvailable: true
		__v: 0
		Error message provided when no book details are provided: Status: 200 OK
		{"message": "BookCopy validation failed: bookDetails : Please add a book details value"}
		Error message provided when an invalid bookDetails id is provided: Status: 200 OK
		{"message": "BookCopy validation failed: bookDetails : Cast to ObjectId failed for value \"63de3c6c8af6473183\" (type string) at path \"bookDetails\" because of \"BSONTypeError\""}
			The information about the book details hasn't been nested. Consider whether this info is likely to be needed on the front end.
			
			Check whether it is okay for a 200 status code to be returned when the bookDetails field is missing or inaccurate.

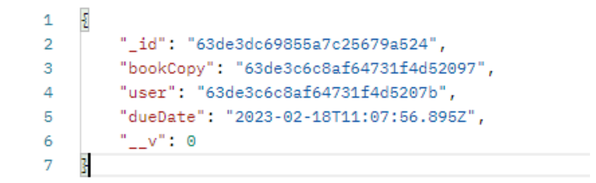
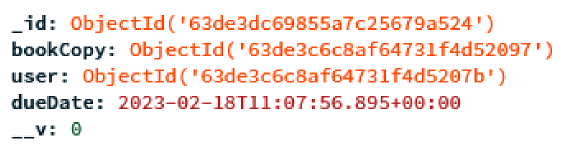
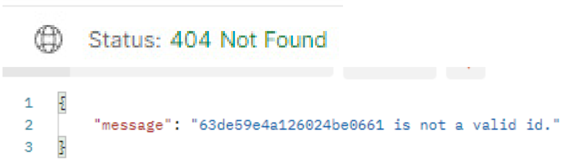
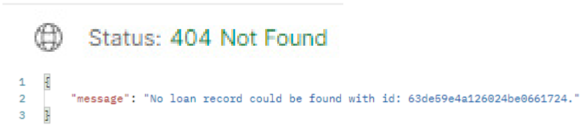
http://localhost:5000/api/bookcopies/:id	**PUT (Update one Book Copy) based on an ID passed in as a URI parameter**
	
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		When details are updated using this route:
		
		`{`
		`"_id": "63de54a06f6a72ebadd48a8e",`
		`"bookDetails": "63de3c6c8af64731f4d52083",`
		`"isAvailable": false,`
		`"__v": 0`
		`}`
		This is then also accurately updated in the DB:
		
		`_id: ObjectId('63de54a06f6a72ebadd48a8e')`
		`bookDetails: ObjectId('63de3c6c8af64731f4d52083')`
		`isAvailable: false`
		`__v: 0`
		When the id isn't valid:  Status: 404 Not Found
		`{"message": "63de3c6c8af64731f4d83 is not a valid id ."}`
		When the id doesn't belong to a bookcopy:
		 Status: 404 Not Found
		`{"message": "No book copy could be found with id: 63 de3c6c8af64731f4d52083."}`
		Invalid token functionality was also tested and found to be working as expected.
			Behaving as expected.

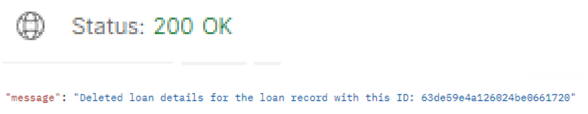
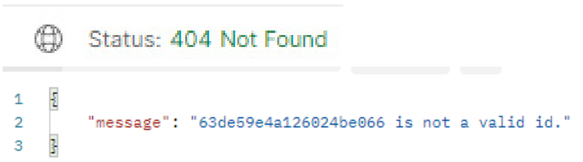
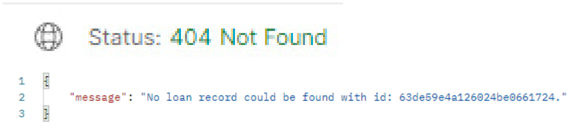
http://localhost:5000/api/bookcopies/:id	**DELETE one Book Copy based on an ID passed in as a URI parameter.**
	
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		When a token and correct id is provided:
		
		`{`
		`"message": "Deleted a book's details with this ID: 63de54a06f6a72ebadd48a8e"`
		`}`
		 Status: 200 OK
		This was then also correctly deleted from the DB which was manually checked in the db.
		When the id isn't valid:  Status: 404 Not Found
		`{"message": "63d8af64731f4d83 is not a valid id."}`
		When the id doesn't belong to a bookcopy:
		 Status: 404 Not Found
		`{"message": "No book copy could be found with id: 63 de3c6c8af64731f4d52094."}`
		Invalid token functionality was also tested and found to be working as expected.
			Behaving as expected

Routes Relating to Loans (FULL CRUD functionality was provided and tested			
End Point	**Expected Result**	**Actual Result**	**Issues Encountered/ Actions to Take**

http://localhost:5000/api/loans	**GET All Loans**
	Return an array of all the current loans. It should:
	• populate the book copy information with details about the book copy including the book details associated with the book
	• Populate the user with their information.
		
		`{`
		`"_id": "63de66162d75e0a6031c9517",`
		`"bookCopy": {`
		`"_id": "63de6493977b13fabd3c7c03",`
		`"bookDetails": {`
		`"_id": "63de6493977b13fabd3c7bf8",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit hol`
		`she can get home. On her journey, Alice is gre`
		`famous tea party with the Mad Hatter and March`
		`"imageList": "https://covers.openlibrary.org/b/isb`
		`"imageDetailed": "https://covers.openlibrary.org/b`
		`"__v": 0`
		`},`
		`"isAvailable": false,`
		`"__v": 0`
		`},`
		`"user": {`
		`"_id": "63de6493977b13fabd3c7bf2",`
		`"username": "Dayle",`
		`"email": "dayleclarke1071@gmail.com",`
		`"password": "$2a$10$WcRR7I0/JLTYF3T8HN/O7OpzUrPsnlTBmD`
		`"isAdmin": true,`
		`"booksOnLoan": [`
		`"63de6493977b13fabd3c7bf8"`
		`],`
		`"__v": 1,`
		`"createdAt": "2023-02-04T13:58:43.581Z",`
		`"updatedAt": "2023-02-04T14:05:10.585Z"`
		`},`
		`"dueDate": "2023-02-18T13:55:08.063Z",`
		`"__v": 0`
		`}`
			Look into removing the password from the output.

http://localhost:5000/api/loans/:id(bookDetails)	**SetLoan: POST a new loan entry.**
	This is expected to do the following:
	• Add an entry to the database to record the loan in the loan collection.
	• Include an automatic due date the is two weeks from today's date.
	• Decrease the available book copies of the book by 1.
	• IsAvailable on the bookCopy will be set to false.
	• Only work if the book has copies currently available.
	• Only work if a valid user id and valid bookDetails id is provided.
	• Add the book to the user's list of books on loan.
		When valid bookDetails and userid are provided:
		Body Cookies Headers (8) Test Results
		Pretty Raw Preview Visualize JSON
		`{`
		`"_id": "63de6493977b13fabd3c7bf2",`
		`"username": "Dayle",`
		`"email": "dayleclarke1071@gmail.com",`
		`"password": "$2a$10$WcRR7I0/JLTYF3T8HN/O7OpzUrPsnlTBmDsLcdzgvSbpD.r1vdBUq"`
		`"isAdmin": true,`
		`"booksOnLoan": [`
		`"63de6493977b13fabd3c7bf8"`
		`],`
		`"__v": 1,`
		`"createdAt": "2023-02-04T13:58:43.581Z",`
		`"updatedAt": "2023-02-04T14:05:10.585Z"`
		`}`
		This information was include in the DB on MongoDB Atlas:
		`_id: ObjectId('63de66162d75e0a6031c9517')`
		`bookCopy: ObjectId('63de6493977b13fabd3c7c03')`
		`user: ObjectId('63de6493977b13fabd3c7bf2')`
		`dueDate: 2023-02-18T13:55:08.063+00:00`
		`__v: 0`
		Available book copies has now decreased by 1
		`{`
		`"_id": "63de6493977b13fabd3c7bf8",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit hole,`
		`she's too big, the next she's too small, and she`
		`constant stream of unlikely characters and situat`
		`caterpillar, and of course, the famous tea party`
		`"imageList": "https://covers.openlibrary.org/b/isbn/9`
		`"imageDetailed": "https://covers.openlibrary.org/b/is`
		`"__v": 0,`
		`"availableCopies": 2`
		`}`
		IsAvailable on the book copy has now been set to false.
		`[`
		`{`
		`"_id": "63de6493977b13fabd3c7c03",`
		`"bookDetails": {`
		`"_id": "63de6493977b13fabd3c7bf8",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit hole`
		`minute she's too big, the next she's too small,`
		`greeted by a constant stream of unlikely charac`
		`conversation with a caterpillar, and of course,`
		`"imageList": "https://covers.openlibrary.org/b/isbr`
		`"imageDetailed": "https://covers.openlibrary.org/b/`
		`"__v": 0`
		`},`
		`"isAvailable": false,`
		`"__v": 0`
		`},`
		It has now been added to the users list of books on loan.
		`{`
		`"id": "63de6493977b13fabd3c7bf2",`
		`"username": "Dayle",`
		`"email": "dayleclarke1071@gmail.com",`
		`"booksOnLoan": [`
		`{`
		`"_id": "63de6493977b13fabd3c7bf8",`
		`"title": "Alice's Adventures in Wonderland",`
		`"author": "Lewis Carroll",`
		`"genre": "Adventure",`
		`"description": "When Alice falls down a rabbit hole, n`
		`minute she's too big, the next she's too small, an`
		`greeted by a constant stream of unlikely character`
		`conversation with a caterpillar, and of course, th`
		`"imageList": "https://covers.openlibrary.org/b/isbn/97`
		`"imageDetailed": "https://covers.openlibrary.org/b/isb`
		`"__v": 0`
		`}`
		`]`
		`}`
			Consider returning more information in the response depending on the requirements of the front end.

http://localhost:5000/api/loans/:id	**PUT (Update one Loan entry) based on an ID passed in as a URI parameter**.
	
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		When a correct token and id is provided:
		
		`{`
		`"_id": "63de3dc69855a7c25679a524",`
		`"bookCopy": "63de3c6c8af64731f4d52097",`
		`"user": "63de3c6c8af64731f4d5207b",`
		`"dueDate": "2023-02-18T11:07:56.895Z",`
		`"__v": 0`
		`}`
		This update was also reflected in the DB.
		
		`_id: ObjectId('63de3dc69855a7c25679a524')`
		`bookCopy: ObjectId('63de3c6c8af64731f4d52097')`
		`user: ObjectId('63de3c6c8af64731f4d5207b')`
		`dueDate: 2023-02-18T11:07:56.895+00:00`
		`__v: 0`
		When the id is not 24 characters:
		
		Status: 404 Not Found
		`{`
		`"message": "63de59e4a126024be0661 is not a valid id."`
		`}`
		When the id doesn't match a loans id
		
		Status: 404 Not Found
		`{`
		`"message": "No loan record could be found with id: 63de59e4a126024be0661724."`
		`}`
		Invalid token functionality was also tested and found to be working as expected.
			More details could be returned to check that this route works.
			This could be populated with the user information and the bookCopy information too.
			
			It might be beneficial to add a route to get one loan entry based on an ID. This would make it easier to check the correct loan is selected before it is deleted. But this would depend on the requirements for the front end.

http://localhost:5000/api/loans/:ID	**DELETE: Loan entry based on an ID passed in as a URI parameter**.
	
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		When a valid token is provided:
		
		Status: 200 OK
		`"message": "Deleted loan details for the loan record with this ID: 63de59e4a126024be0661720"`
		This was then also correctly deleted from the DB which was manually checked in the db.
		
		When the id is not 24 characters:
		
		Status: 404 Not Found
		`{`
		`"message": "63de59e4a126024be066 is not a valid id."`
		`}`
		When the id doesn't match a loans id
		
		Status: 404 Not Found
		`{`
		`"message": "No loan record could be found with id: 63de59e4a126024be0661724."`
		`}`
		Invalid token functionality was also tested and found to be working as expected.
			Behaving as expected

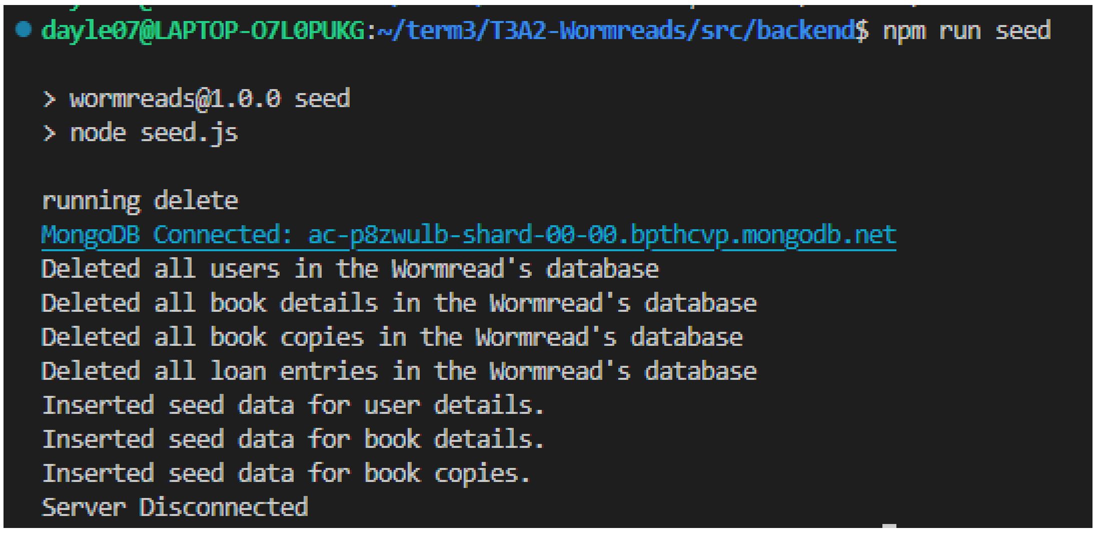
Manual testing of seeding the database:

Expected Result:

All entries, in all collections on the database stored on MongoDB Atlas are deleted. The seed file then populates the database with mock data in each collection. The database will connect at the beginning of the script and disconnect at the end of the script. Useful messages are logged to the console to inform the user what is happening.

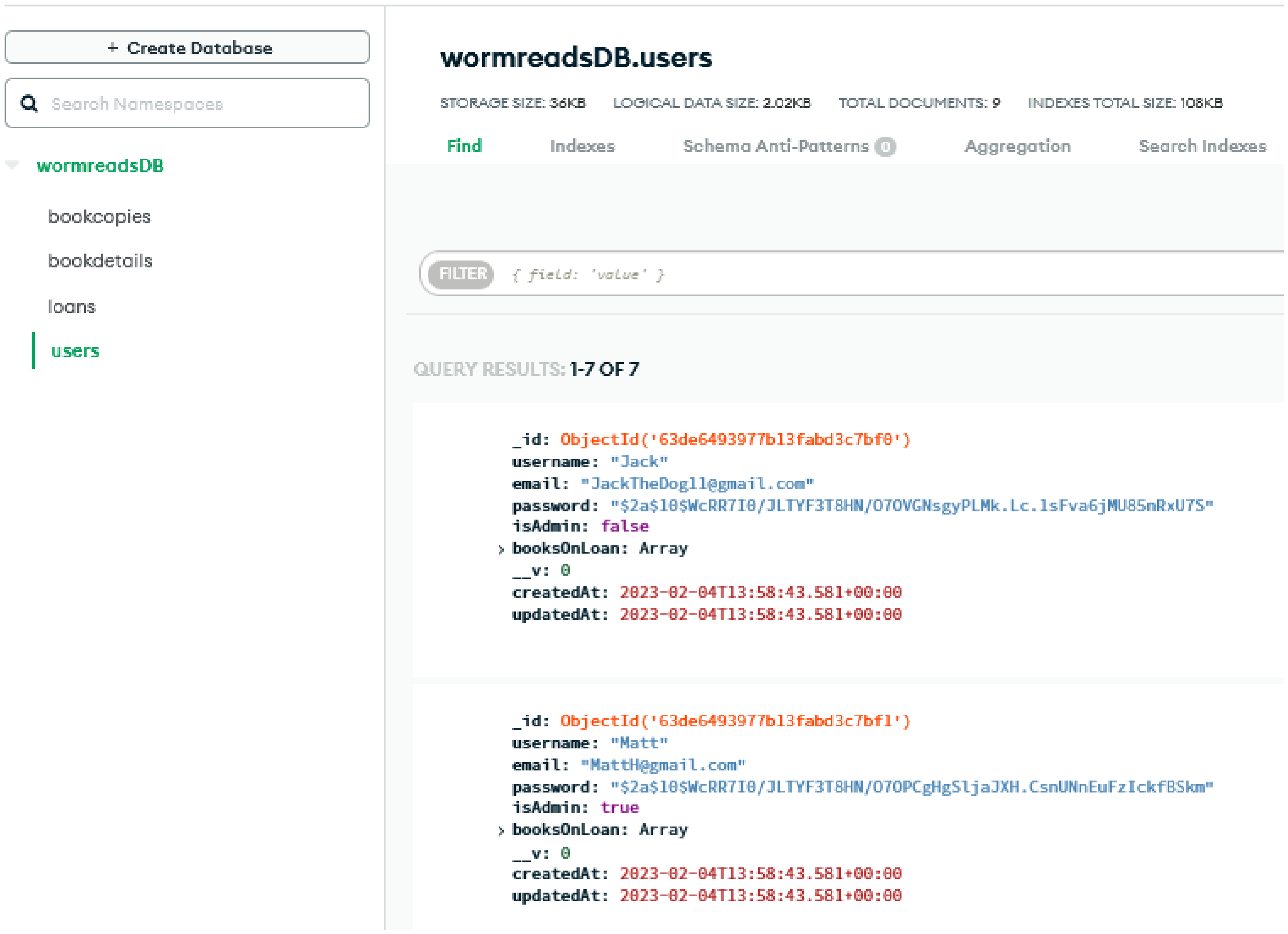
Actual Result:

The following messages were logged to the console as expected:



```
dayle07@LAPTOP-O7L0PUKG:~/term3/T3A2-Wormreads/src/backend$ npm run seed

> wormreads@1.0.0 seed
> node seed.js

running delete
MongoDB Connected: ac-p8zwulb-shard-00-00.bpthcvp.mongodb.net
Deleted all users in the Wormread's database
Deleted all book details in the Wormread's database
Deleted all book copies in the Wormread's database
Deleted all loan entries in the Wormread's database
Inserted seed data for user details.
Inserted seed data for book details.
Inserted seed data for book copies.
Server Disconnected
```

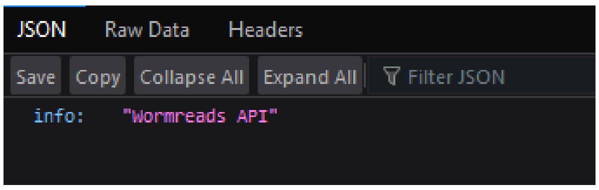
All need data was populated in MondoDB Atlas as expected. For example, the user's collection appears as follows:

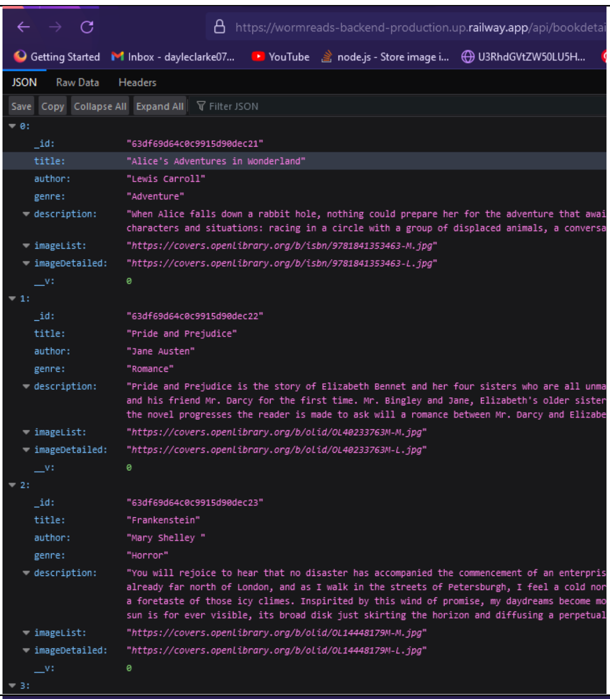


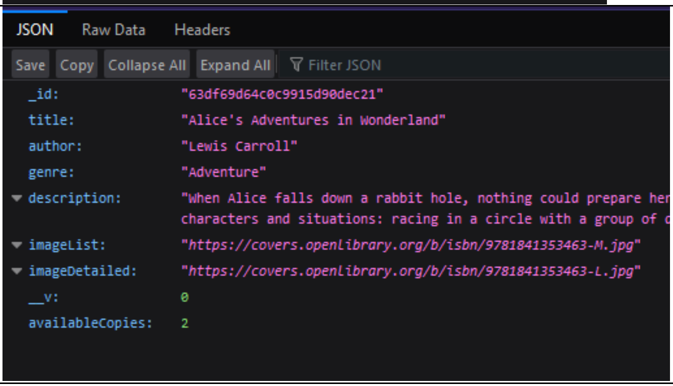
```
_id: ObjectId('63de6493977b13fabd3c7bf0')
username: "Jack"
email: "JackTheDog11@gmail.com"
password: "$2a$10$WcRR7I0/JLTYF3T8HN/O7OVGNsgyPLMk.Lc.1sFva6jMU85nRxU7S"
isAdmin: false
> booksOnLoan: Array
__v: 0
createdAt: 2023-02-04T13:58:43.581+00:00
updatedAt: 2023-02-04T13:58:43.581+00:00

_id: ObjectId('63de6493977b13fabd3c7bf1')
username: "Matt"
email: "MattH@gmail.com"
password: "$2a$10$WcRR7I0/JLTYF3T8HN/O7OPCgHgSljaJXH.CsnUNnEuFzIckfBSkm"
isAdmin: true
> booksOnLoan: Array
__v: 0
createdAt: 2023-02-04T13:58:43.581+00:00
updatedAt: 2023-02-04T13:58:43.581+00:00
```

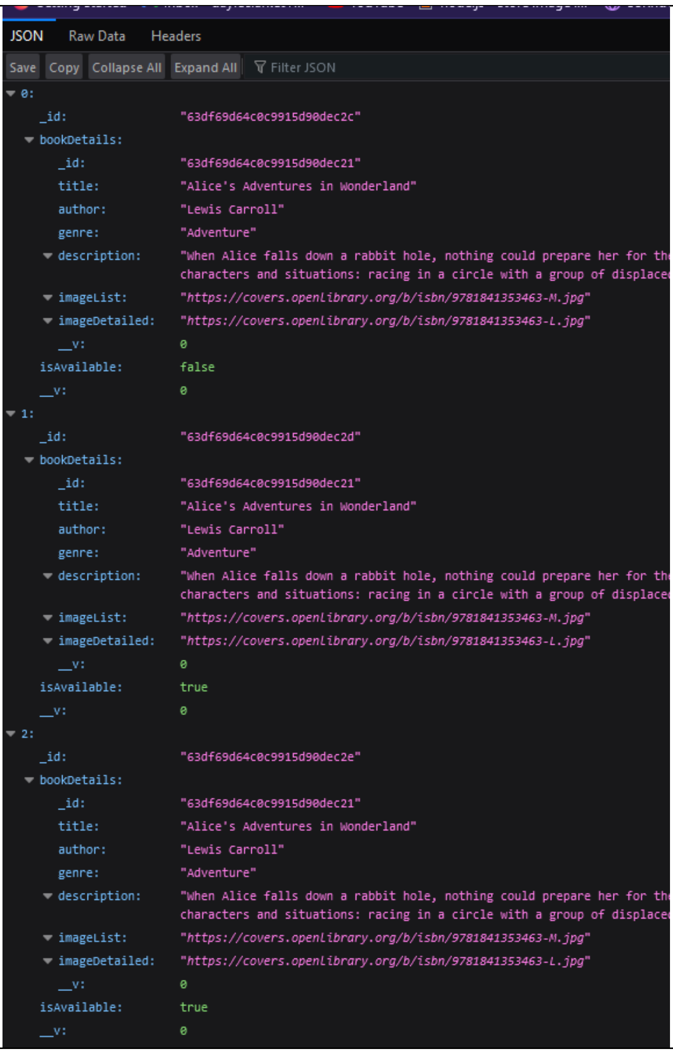
User-Testing of Back-End Production Site

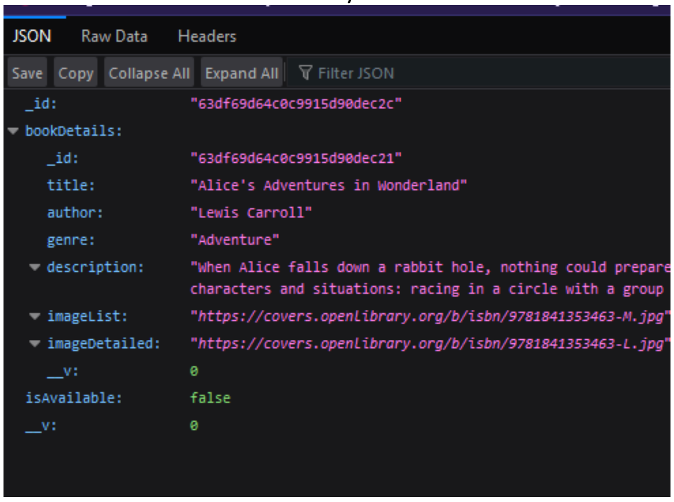
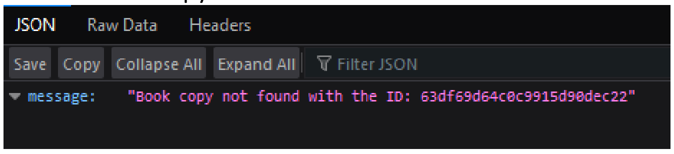
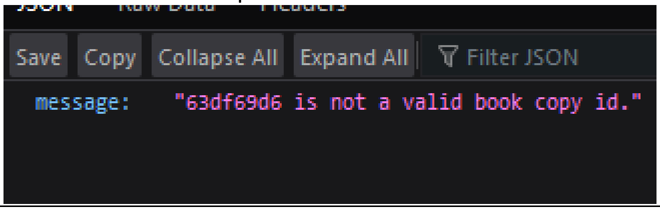
The following manual testing was run using Postman to test to endpoints of the API after it was deployed to Railway and used in production .

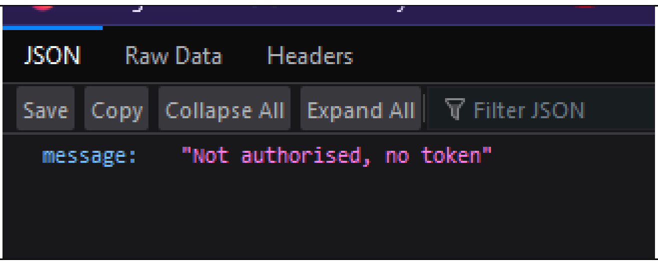
End Point	Expected Result	Actual Result	Issues Encountered/ Actions to Take
https://wormreads-backend-production.up.railway.app/	**Homepage** Should display in JSON and return a message stating Wormreads API. Return a 200 status code.	 JSON Raw Data Headers Save Copy Collapse All Expand All Filter JSON `info: "Wormreads API"`	Behaving as expected

https://wormreads-backend-production.up.railway.app/api/bookdetails	**GET All BookDetails** (no token required)
	Should display a list of all book details in an array with the following fields:
	• Id (autogenerated, 24 characters))
	• Title
	• Author
	• Genre
	• Description
	• imageList
	• imagedetails
	Should have a 200 status code
	Response should be in JSON
	Return a 200 status code.
		
		https://wormreads-backend-production.up.railway.app/api/bookdetai
		Getting Started Inbox - dayleclarke07... YouTube node.js - Store image i... U3RhdGVtZW50LU5H...
		JSON Raw Data Headers
		Save Copy Collapse All Expand All Filter JSON
		`0:`
		`_id: "63df69d64c0c9915d90dec21"`
		`title: "Alice's Adventures in Wonderland"`
		`author: "Lewis Carroll"`
		`genre: "Adventure"`
		`description: "When Alice falls down a rabbit hole, nothing could prepare her for the adventure that awai characters and situations: racing in a circle with a group of displaced animals, a conversa`
		`imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg"`
		`imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg"`
		`__v: 0`
		`1:`
		`_id: "63df69d64c0c9915d90dec22"`
		`title: "Pride and Prejudice"`
		`author: "Jane Austen"`
		`genre: "Romance"`
		`description: "Pride and Prejudice is the story of Elizabeth Bennet and her four sisters who are all unma and his friend Mr. Darcy for the first time. Mr. Bingley and Jane, Elizabeth's older sister the novel progresses the reader is made to ask will a romance between Mr. Darcy and Elizabe`
		`imageList: "https://covers.openlibrary.org/b/olid/OL40233763M-M.jpg"`
		`imageDetailed: "https://covers.openlibrary.org/b/olid/OL40233763M-L.jpg"`
		`__v: 0`
		`2:`
		`_id: "63df69d64c0c9915d90dec23"`
		`title: "Frankenstein"`
		`author: "Mary Shelley "`
		`genre: "Horror"`
		`description: "You will rejoice to hear that no disaster has accompanied the commencement of an enterpris already far north of London, and as I walk in the streets of Petersburgh, I feel a cold nor a foretaste of those icy climes. Inspirited by this wind of promise, my daydreams become mo sun is for ever visible, its broad disk just skirting the horizon and diffusing a perpetual`
		`imageList: "https://covers.openlibrary.org/b/olid/OL14448179M-M.jpg"`
		`imageDetailed: "https://covers.openlibrary.org/b/olid/OL14448179M-L.jpg"`
		`__v: 0`
		`3:`
			Behaving as expected

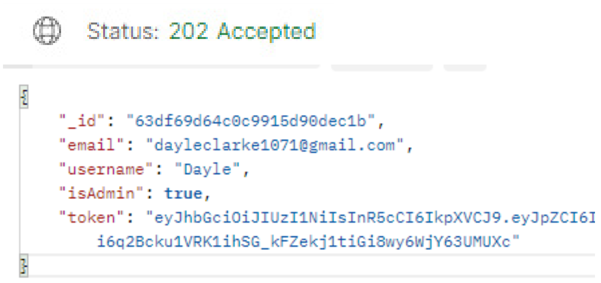
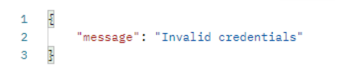
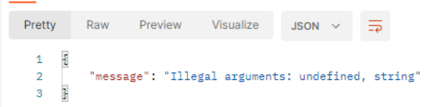
https://wormreads-backend-production.up.railway.app/api/bookdetails/63df69d64c0c9915d90dec21	**GET One BookDetails** (no token required)
	Should display a single book with all of its details (as above) based on the Book Details ID provided.
	Response should be in JSON
	Return a 200 status code.
		
		JSON Raw Data Headers
		Save Copy Collapse All Expand All Filter JSON
		`_id: "63df69d64c0c9915d90dec21"`
		`title: "Alice's Adventures in Wonderland"`
		`author: "Lewis Carroll"`
		`genre: "Adventure"`
		`description: "When Alice falls down a rabbit hole, nothing could prepare her characters and situations: racing in a circle with a group of`
		`imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg"`
		`imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg"`
		`__v: 0`
		`availableCopies: 2`
			Behaving as expected

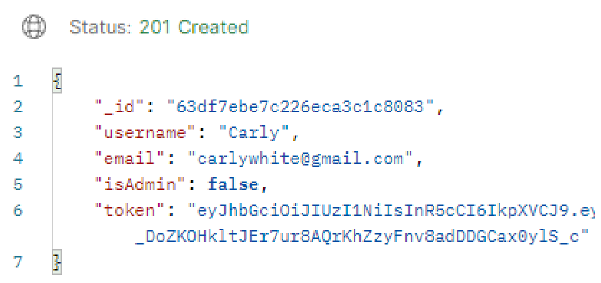
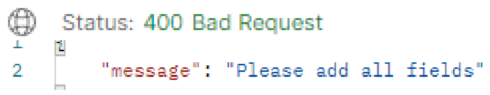
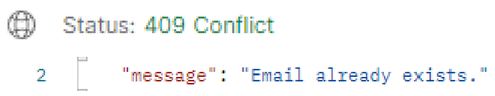
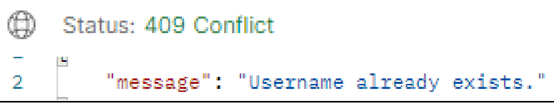
https://wormreads-backend-production.up.railway.app/api/bookcopies	**Get All Book Copies**
	Should return an array containing all the bookCopies stored in the DB. It should display the book copies id, a Boolean indicating whether the book copy is available and a nested bookDetails object with details about the book.
	Response should be in JSON
	Return a 200 status code.
		
		JSON Raw Data Headers
		Save Copy Collapse All Expand All Filter JSON
		▾ 0:
		  _id: "63df69d64c0c9915d90dec2c"
		  ▾ bookDetails:
		    _id: "63df69d64c0c9915d90dec21"
		    title: "Alice's Adventures in Wonderland"
		    author: "Lewis Carroll"
		    genre: "Adventure"
		    ▾ description: "When Alice falls down a rabbit hole, nothing could prepare her for th characters and situations: racing in a circle with a group of displace
		    ▾ imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg"
		    ▾ imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg"
		    __v: 0
		  isAvailable: false
		  __v: 0
		▾ 1:
		  _id: "63df69d64c0c9915d90dec2d"
		  ▾ bookDetails:
		    _id: "63df69d64c0c9915d90dec21"
		    title: "Alice's Adventures in Wonderland"
		    author: "Lewis Carroll"
		    genre: "Adventure"
		    ▾ description: "When Alice falls down a rabbit hole, nothing could prepare her for th characters and situations: racing in a circle with a group of displace
		    ▾ imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg"
		    ▾ imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg"
		    __v: 0
		  isAvailable: true
		  __v: 0
		▾ 2:
		  _id: "63df69d64c0c9915d90dec2e"
		  ▾ bookDetails:
		    _id: "63df69d64c0c9915d90dec21"
		    title: "Alice's Adventures in Wonderland"
		    author: "Lewis Carroll"
		    genre: "Adventure"
		    ▾ description: "When Alice falls down a rabbit hole, nothing could prepare her for th characters and situations: racing in a circle with a group of displace
		    ▾ imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg"
		    ▾ imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg"
		    __v: 0
		  isAvailable: true
		  __v: 0
			Behaving as expected

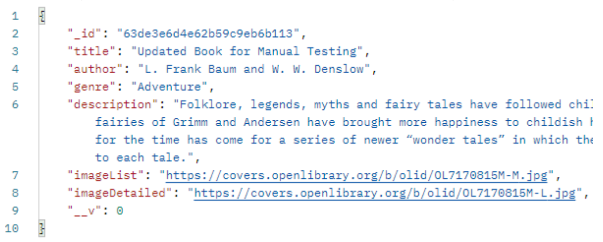
https://wormreads-backend-production.up.railway.app/api/bookcopies/63df69d64c0c9915d90dec2c	**GET a single book copy based on their id.** • Provide error messages if the id provided is not valid. • Response should be in JSON • Return a 200 status code.	When the ID is entered correctly:  JSON Raw Data Headers Save Copy Collapse All Expand All Filter JSON _id: "63df69d64c0c9915d90dec2c" ▾ bookDetails: _id: "63df69d64c0c9915d90dec21" title: "Alice's Adventures in Wonderland" author: "Lewis Carroll" genre: "Adventure" ▾ description: "When Alice falls down a rabbit hole, nothing could prepare characters and situations: racing in a circle with a group ▾ imageList: "https://covers.openlibrary.org/b/isbn/9781841353463-M.jpg" ▾ imageDetailed: "https://covers.openlibrary.org/b/isbn/9781841353463-L.jpg" __v: 0 isAvailable: false __v: 0 When an correctly formatted ID is provided that doesn't match a book copy in the DB:  JSON Raw Data Headers Save Copy Collapse All Expand All Filter JSON ▾ message: "Book copy not found with the ID: 63df69d64c0c9915d90dec22" When an invalid ID is provided:  Save Copy Collapse All Expand All Filter JSON message: "63df69d6 is not a valid book copy id."	Behaving as expected

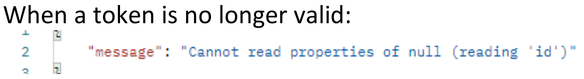
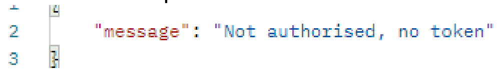
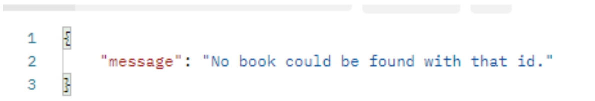
https://wormreads-backend-production.up.railway.app/api/users	**Get all Users (without a token)** Should return a not authorised no token message and a 401Unauthorized	`message: "Not authorised, no token"`	Behaving as Expected

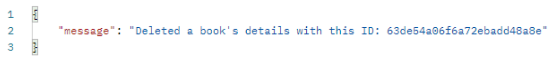
Routes checked in Postman After Deployment

https://wormreads-backend-production.up.railway.app/api/users/login	**POST- Login a User**
	When a correct username and password are passed in as a JSON body it should return the user's credentials and a JWT token. Should have a 202 status code.
	When incorrect details are provided an error message should be displayed in JSON indicating that their credentials are incorrect
		When correct details are provided:
		Status: 202 Accepted
		`{`
		`"_id": "63df69d64c0c9915d90dec1b",`
		`"email": "dayleclarke1071@gmail.com",`
		`"username": "Dayle",`
		`"isAdmin": true,`
		`"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6I`
		`i6q2Bcku1VRK1ihSG_kFZekj1tiGi8wy6WjY63UMUXc"`
		`}`
		
		When incorrect details are passed in:
		`{`
		`"message": "Invalid credentials"`
		`}`
		When either the username or password is missing:
		`{`
		`"message": "Illegal arguments: undefined, string"`
		`}`
			Behaving as Expected

https://wormreads-backend-production.up.railway.app/api/users/register	**POST- Register a User**
	When a unique username and email are provided along with a password a new user is created and added to the users collection in the database.
	An error message with a 409 conflict error should be provided if a username or email has already been taken and the message should specifically state which one is missing.
	An error message should be displayed if a username, email or password is missing.
		When all the correct fields are provided:
		Status: 201 Created
		`{`
		`"_id": "63df7ebe7c226eca3c1c8083",`
		`"username": "Carly",`
		`"email": "carlywhite@gmail.com",`
		`"isAdmin": false,`
		`"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey`
		`_DoZKOHkltJEr7ur8AQrKhZzyFnv8adDDGCax0ylS_c"`
		`}`
		The password has also been added to the database and hashed. This is shown on MongoDB Atlas.
		
		When a field is missing:
		Status: 400 Bad Request
		`"message": "Please add all fields"`
		When the email is already taken:
		Status: 409 Conflict
		`"message": "Email already exists."`
		When the username is already taken:
		Status: 409 Conflict
		`"message": "Username already exists."`	

https://wormreads-backend-production.up.railway.	**Update (PUT) One BookDetails Entry**
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		Put request when one or more fields are provided:
		`{`
		`"_id": "63de3e6d4e62b59c9eb6b113",`
		`"title": "Updated Book for Manual Testing",`
		`"author": "L. Frank Baum and W. W. Denslow",`
		`"genre": "Adventure",`
		`"description": "Folklore, legends, myths and fairy tales have followed chi`
		`fairies of Grimm and Andersen have brought more happiness to childish`
		`for the time has come for a series of newer “wonder tales” in which the`
		`to each tale.",`
		`"imageList": "https://covers.openlibrary.org/b/olid/OL7170815M-M.jpg",`
		`"imageDetailed": "https://covers.openlibrary.org/b/olid/OL7170815M-L.jpg",`
		`"__v": 0`
		`}`	

		When a token is no longer valid:
		
		"message": "Cannot read properties of null (reading 'id')"
		When no token is provided:
		
		{
		"message": "Not authorised, no token"
		}
		Error message thrown when a URI is passed in that isn't 24 character long:
		"message": "Cast to ObjectId failed for value \"63de3e6d4e62b59c9eb6b14\" (type string) at path \"_id\" for model \"BookDetails\""
		Error message thrown when an id provided but it is invalid:
		
		{
		"message": "No book could be found with that id."
		}	

https://wormreads-backend-production.up.railway.app/api/bookcopies/63df69d64c0c9915d90dec2c	**DELETE one Book Copy based on an ID passed in as a URI parameter.**
	
	These changes should be reflected in the actual DB. Error messages should be provided if a token is not provided from an admin user or the ID passed in is not valid.
		When a token and correct id is provided:
		
		{
		"message": "Deleted a book's details with this ID: 63de54a06f6a72ebadd48a8e"
		}
		
		Status: 200 OK
		This was then also correctly deleted from the DB which was manually checked in the db.
		When the id isn't valid: Status: 404 Not Found
		{"message": "63d8af64731f4d83 is not a valid id."}
		
		When the id doesn't belong to a bookcopy:
		
		Status: 404 Not Found
		{"message": "No book copy could be found with id: 63de3c6c8af64731f4d52094."}
		Invalid token functionality was also tested and found to be working as expected.
			Behaving as expected