

1. Testing	2
1.1 Acceptance test for sprint 1	3
1.2 Acceptance test for sprint 2	4
1.3 Database Unit Test for sprint 1	5
1.4 Database Unit Test for sprint 2	13
1.5 Integration test for Sprint 1	15
1.6 Integration test for Sprint 2	20
1.7 Schema validation test	21

Testing

Acceptance test for sprint 1

MongoDB							
Function	Result	Output					
Deployed Back-end on Oracle cloud	PASS	<pre>rsrepo.service - RS Repo Backend Loaded: loaded (/etc/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Tue 2021-05-04 12:27:05 UTC; 13h ago Main PID: 889934 (node) Tasks: 11 (limit: 1110) Memory: 74.6M CGroup: /system.slice/rsrepo.service └─889934 /usr/bin/node index.js</pre>					
Deployed database on MongoDB cloud	PASS	mongodb+srv://root:root@cluster0.uhlwx.mongodb.net/myFirstDatabase?retryWrites=true&w=majority					
collection data	PASS	Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
		articles	32	284.2 B	8.9 KB	4	128.0 KB
		bookmarks	0	-	0.0 B	1	4.0 KB
		comments	12	267.6 B	3.1 KB	1	36.0 KB
		pendingarticles	12	185.9 B	2.2 KB	1	36.0 KB
		sections	75	132.1 B	9.7 KB	1	36.0 KB
		subjects	18	158.4 B	2.8 KB	3	108.0 KB
		tools	2	73.5 B	147.0 B	2	72.0 KB
		users	20	296.5 B	5.8 KB	3	108.0 KB
articles structure	PASS	<pre>/** * Paste one or more documents here */ { "subjects": ["SUBJECT"], "tools": null, "tags": ["SUBJECT", "ttt"], "is_pending": false, "likes": [], "create_at": { "\$date": "2021-03-18T11:22:44.348Z" }, "title": "article", "author_id": { "\$oid": "608a7f3e07d4435f632cb72c" }, "content": "new articlesafdafgqwaegegaegwgg", "comment_section": { "\$oid": "6053380455631157f25addf2" }, "__v": 0 }</pre>					

Acceptance test for sprint 2

Redis Cache		
Function	Result	Output
redis server installation	PASS	<div>Reading package lists... Done Building dependency tree Reading state information... Done The following additional packages will be installed: binutils binutils-common binutils-x86-64-linux-gnu bu fonts-dejavu-core g++ g++-9 gcc gcc-9 gcc-9-base gyp libalgorithm-diff-xs-perl libalgorithm-merge-perl lib libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-n libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libencod libfile-desktopentry-perl libfile-fcntllock-perl libf libfontconfig1 libfontenc1 libgcc-9-dev libgl1 libgl1 libhtml-form-perl libhtml-format-perl libhtml-parser- libhttp-daemon-perl libhttp-date-perl libhttp-message libio-socket-ssl-perl libio-stringy-perl libipc-syste libjs-is-typedarray libjs-psl libjs-typedarray-to-buf liblwp-protocol-https-perl libmailtools-perl libmpc3 libnet-ssleay-perl libnode-dev libnode64 libpciaccess libquadmath0 libsensors-config libsensors5 libsm6 lib</div> <div>Setting up node-http-signature (1.3.2-1) ... Setting up node-fs-vacuum (1.2.10-3) ... Setting up node-gauge (2.7.4-1) ... Setting up node-wrap-ansi (4.0.0-2) ... Setting up node-normalize-package-data (2.5.0-1) ... Setting up node-bson (4.2.0-2) ... Setting up node-package-json (4.0.1-1) ... Setting up node-latest-version (3.1.0-1) ... Setting up node-request (2.88.1-4) ... Setting up node-memfs (4.1.2-2) ... Setting up node-cliui (4.1.0-2) ... Setting up node-cacache (11.3.3-2) ... Setting up node-read-package-json (2.1.1-1) ... Setting up node-gyp (6.1.0-3) ... Setting up node-libmax (10.2.1-2) ... Setting up npm (6.14.4+ds-1ubuntu2) ... Setting up libwww-perl (6.43-1) ... Setting up liblwp-protocol-https-perl (6.07-2ubuntu2) ... Setting up libhtml-parser-perl (2.46-1) ... Setting up libhtml-twig-perl (1:3.50-2) ... Setting up libnet-dbus-perl (1.2.0-1) ... Processing triggers for systemd (245.4-ubuntu3.6) ... Processing triggers for man-db (2.9.1-1) ... Processing triggers for mime-support (3.64ubuntu1) ... Processing triggers for libc-bin (2.31-0ubuntu9.2) ...</div>
redis connection	PASS	[winston] Attempt to write logs with no transports {"message":"redisCache connection succeed","level":"info"}
cache lifetime	PASS	[winston] Attempt to write logs with no transports {"message":"redisCache restart","level":"info"}

Database Unit Test for sprint 1

Since the database is a crucial part of the project, the Unit Test is used for testing Database functions such as insert, search, update, and delete to avoid any unexpected result.

Users documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search an admin user with the account name called "unit_test"	db.users.find({account: "unit_test"})	Return nothing	Return nothing	PASS	
2	insert an admin user	db.users.insertOne({ "is_moderator": true, "is_admin": true, "subscribed_tools": [], "subscribed_subjects": ["COMP90013"], "moderated_subjects": ["COMP90013"], "articles": ["Java"], "images": [], "account": "unit_test", "password": "\$2b\$10\$/Dz5oh3ttr1j6r6/ISHRMOsHbnV4YYPYAet9CNO0XlqAr679zG0.K", "student_number": "000000", })	{ acknowledged: true, insertedId: ObjectId("608eac785c58d53264afe583") }	{ acknowledged: true, insertedId: ObjectId("608eac785c58d53264afe583") }	PASS	It's also PASS for inserting a student account.
3	search an admin user with the account name called "unit_test"	db.users.find({account: "unit_test"})	{ _id: ObjectId("608eac785c58d53264afe583"), is_moderator: true, is_admin: true, subscribed_tools: [], subscribed_subjects: ['COMP90013'], moderated_subjects: ['COMP90013'], articles: ['Java'], images: [], account: 'unit_test', password: '\$2b\$10\$/Dz5oh3ttr1j6r6/ISHRMOsHbnV4YYPYAet9CNO0XlqAr679zG0.K', student_number: '000000' }	{ _id: ObjectId("608eac785c58d53264afe583"), is_moderator: true, is_admin: true, subscribed_tools: [], subscribed_subjects: ['COMP90013'], moderated_subjects: ['COMP90013'], articles: ['Java'], images: [], account: 'unit_test', password: '\$2b\$10\$/Dz5oh3ttr1j6r6/ISHRMOsHbnV4YYPYAet9CNO0XlqAr679zG0.K', student_number: '000000' }	PASS	It's also PASS for searching the student account.

4	update the subscribed subjects for the user of "unit_test" from 'COMP90013' to 'COMP90008'	db.users.updateOne ({account: "unit_test"}, {\$set: {subscribed_subjects: ['COMP90008']}}) db.users.find ({account: "unit_test"})	{ _id: ObjectId ("608eac785c58d53264afe583"), is_moderator: true, is_admin: true, subscribed_tools: [], moderated_subjects: ['COMP90008'], articles: ['Java'], images: [], account: 'unit_test', password: '\$2b\$10\$/Dz5oh3ttrlj6r6/ISHRMOsHbnV4YYPYAet9CNO0XlqAr679zG0.K', student_number: '000000' }	PASS	It's also PASS for if the updated account is a student account.	
5	delete the user account of "unit_test"	db.users.deleteOne ({account: "unit_test"}) db.users.find ({account: "unit_test"})	Return nothing when trying to find the "unit_test" account after deletion	Return nothing when trying to find the "unit_test" account after deletion	PASS	It's also PASS if the deletion is a student account.
6	insert a user account with an empty document	db.users.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a record only with a password	db.users.insertOne ({password: "unit_test"}) db.users.find ({password: "unit_test"})	{ _id: ObjectId ("608fb3e05c58d53264afe592"), password: 'unit_test' }	Error message saying missing value for the account	FAIL	should make the account the mandatory field
8	insert a user with an account only	db.users.insertOne ({account: "unit_test"}) db.users.find ({account: "unit_test"})	{ _id: ObjectId ("608fb5d55c58d53264afe593"), account: 'unit_test' }	Error message saying missing value for the password	FAIL	should make the password the mandatory field
9	insert a user with a student_number that exceeds 8 numbers	db.users.insertOne ({account: "unit_test", password: "unit_test", student_number: "123456789"}) db.users.find ({account: "unit_test"})	{ _id: ObjectId ("608fb6835c58d53264afe596"), account: 'unit_test', password: 'unit_test', student_number: '123456789' }	Error message saying student_number should be digits only	FAIL	should limit the student_number to be digits and 6 to 8 numbers only
10	insert a user with the password of test when there is already a user with a password of test	db.users.insertOne ({account: "unit_test", password: "test", student_number: "123456"}) db.users.find ({account: "unit_test"})	{ _id: ObjectId ("608fb6835c58d53264afe596"), account: 'unit_test', password: 'test', student_number: '123456' }	MongoError: E11000 duplicate key error collection: myFirstDatabase.users index: student_number_1 dup key: { student_number: "test" }	FAIL	should make the student_number as the key rather than the password

ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search a tool with the tool name called "unit_test"	db.tools.find({"name":"unit_test"})	Return nothing	Return nothing	PASS	
2	insert a tool with the name called "unit_test"	db.tools.insertOne({"articles":["Github", "Java", "JS"],"name":"unit_test"})	{ acknowledged: true, insertedId: ObjectId("608eb63d5c58d53264afe584") }	{ acknowledged: true, insertedId: ObjectId("608eb63d5c58d53264afe584") }	PASS	
3	search a tool with the tool name called "unit_test"	db.tools.find({"name":"unit_test"})	{ _id: ObjectId("608eb63d5c58d53264afe584"), articles: ['Github', 'Java', 'JS'], name: 'unit_test' }	{ _id: ObjectId("608eb63d5c58d53264afe584"), articles: ['Github', 'Java', 'JS'], name: 'unit_test' }	PASS	
4	update a tool's articles from ['Github', 'Java', 'JS'] to ['Github', 'JS']	db.tools.updateOne({name:"unit_test"},{\$set:{"articles":["Github", "JS"]}}) db.tools.find({name: "unit_test"})	{ _id: ObjectId("608eb63d5c58d53264afe584"), articles: ['Github', 'JS'], name: 'unit_test' }	{ _id: ObjectId("608eb63d5c58d53264afe584"), articles: ['Github', 'JS'], name: 'unit_test' }	PASS	
5	delete the document of "unit_test"	db.tools.deleteOne({name:"unit_test"}) db.tools.find({name: "unit_test"})	Return nothing	Return nothing	PASS	
6	insert a tool with an empty document	db.tools.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a tool with articles only	db.tools.insertOne({articles:"unit_test"})	{ acknowledged: true, insertedId: ObjectId("608fb3415c58d53264afe591") }	Error message saying missing value for name	FAIL	should make the name the mandatory field

Subjects documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search a subject with the name called "unit_test"	db.subjects.find({"name":"unit_test"})	Return nothing	Return nothing	PASS	
2	insert a subject with the name called "unit_test"	db.subjects.insertOne({"subject_code":"COMP90000","name":"unit_test", "description":"testing the subject for unit test"})	{ acknowledged: true, insertedId: ObjectId("608fa1b75c58d53264afe585") }	{ acknowledged: true, insertedId: ObjectId("608fa1b75c58d53264afe585") }	PASS	

3	search a subject with the name called "unit_test"	db.subjects.find({ "name": "unit_test" })	{ _id: ObjectId("608fa1b75c58d53264afe585"), subject_code: 'COMP90000', name: 'unit_test', description: 'testing the subject for unit test' }	{ _id: ObjectId("608fa1b75c58d53264afe585"), subject_code: 'COMP90000', name: 'unit_test', description: 'testing the subject for unit test' }	PASS	
4	update a subject's code from 'COMP90000' to 'COMP90001'	db.subjects.updateOne({name: "unit_test"}, {\$set: {"subject_code": "COMP90001"}}) db.subjects.find({name: "unit_test"})	{ _id: ObjectId("608fa1b75c58d53264afe585"), subject_code: 'COMP90001', name: 'unit_test', description: 'testing the subject for unit test' }	{ _id: ObjectId("608fa1b75c58d53264afe585"), subject_code: 'COMP90001', name: 'unit_test', description: 'testing the subject for unit test' }	PASS	
5	delete the document of "unit_test"	db.subjects.deleteOne({name: "unit_test"}) db.subjects.find({name: "unit_test"})	Return nothing	Return nothing	PASS	
6	insert a subject with an empty document	db.subjects.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a subject with a name but an empty value for subject_code and description	db.subjects.insertOne({name: "unit_test"})	{ acknowledged: true, insertedId: ObjectId("608fa3d55c58d53264afe586") }	Error message saying missing value of subject_code and created_by	FAIL	should at least make subject_code, and created_by mandatory
8	insert a subject with a description but an empty value for subject_code and name	db.subjects.insertOne({description: "unit_test"})	{ acknowledged: true, insertedId: ObjectId("608fa5555c58d53264afe587") }	Error message saying missing value of subject_code, name, and created_by	FAIL	should at least make subject_code, name, and created_by mandatory

Sections documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search a section with the name called "unit_test"	db.sections.find({"name": "unit_test"})	Return nothing	Return nothing	PASS	

2	insert a section with the name called "unit_test"	db.sections.insertOne({ "owner": "COMP90000", "name": "unit_test", "comments": "testing the subject for unit test" })	{ acknowledged: true, insertedId: ObjectId("608fa6f05c58d53264afe588") }	{ acknowledged: true, insertedId: ObjectId("608fa6f05c58d53264afe588") }	PASS	
3	search a section with the name called "unit_test"	db.sections.find({ "name": "unit_test" })	{ _id: ObjectId("608fa6f05c58d53264afe588"), owner: 'COMP90000', name: 'unit_test', comments: 'testing the subject for unit test' }	{ _id: ObjectId("608fa6f05c58d53264afe588"), owner: 'COMP90000', name: 'unit_test', comments: 'testing the subject for unit test' }	PASS	
4	update a section's owner from 'COMP90000' to 'COMP90001'	db.sections.updateOne({ name: "unit_test", \$set: { "owner": "COMP90001" } }) db.sections.find({ name: "unit_test" })	{ _id: ObjectId("608fa6f05c58d53264afe588"), owner: 'COMP90001', name: 'unit_test', comments: 'testing the subject for unit test' }	{ _id: ObjectId("608fa6f05c58d53264afe588"), owner: 'COMP90001', name: 'unit_test', comments: 'testing the subject for unit test' }	PASS	
5	delete the document of "unit_test"	db.sections.deleteOne({ name: "unit_test" }) db.sections.find({ name: "unit_test" })	Return nothing	Return nothing	PASS	
6	insert a section with an empty document	db.sections.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a section with the comments only	db.sections.insertOne({ comments: "unit_test" })	{ acknowledged: true, insertedId: ObjectId("608fa8315c58d53264afe58a") }	Error message saying missing value for name, owner, type, and articles	FAIL	should at least make the name, owner, type, and articles as the mandatory fields

Pendingarticles documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search a pending article with the title called "unit_test"	db.pendingarticles.find({"title": "unit_test"})	Return nothing	Return nothing	PASS	
2	insert a pending article with the title called "unit_test"	db.pendingarticles.insertOne({ "subjects": "COMP90000", "title": "unit_test", "content": "testing the subject for unit test" })	{ acknowledged: true, insertedId: ObjectId("608fa995c58d53264afe58b") }	{ acknowledged: true, insertedId: ObjectId("608fa995c58d53264afe58b") }	PASS	

3	search a pending article with the title called "unit_test"	db.pendingarticles.find({"title": "unit_test"})	<pre>{ _id: ObjectId("608fa9955c58d53264afe58b"), subjects: 'COMP90000', title: 'unit_test', content: 'testing the subject for unit test' }</pre>	<pre>{ _id: ObjectId("608fa9955c58d53264afe58b"), subjects: 'COMP90000', title: 'unit_test', content: 'testing the subject for unit test' }</pre>	PASS	
4	update a pending article's subjects from 'COMP90000' to 'COMP90001'	db.pendingarticles.updateOne({title: "unit_test"}, {\$set: {"subjects": "COMP90001"}}) db.pendingarticles.find({title: "unit_test"})	<pre>{ _id: ObjectId("608fa9955c58d53264afe58b"), subjects: 'COMP90001', title: 'unit_test', content: 'testing the subject for unit test' }</pre>	<pre>{ _id: ObjectId("608fa9955c58d53264afe58b"), subjects: 'COMP90001', title: 'unit_test', content: 'testing the subject for unit test' }</pre>	PASS	
5	delete the document of "unit_test"	db.pendingarticles.deleteOne({title: "unit_test"}) db.pendingarticles.find({title: "unit_test"})	Return nothing	Return nothing	PASS	
6	insert a pending article with an empty document	db.pendingarticles.insertOne()	MongoshInvalidInputError : Missing required argument	MongoshInvalidInputError : Missing required argument	PASS	
7	insert a pending article with the content only	db.pendingarticles.insertOne({content: "unit_test"})	<pre>{ acknowledged: true, insertedId: ObjectId("608faaa15c58d53264afe58c") }</pre>	Error message saying missing value for subjects, title, editor_id etc	FAIL	should at least make the subjects, title, editor_id as the mandatory fields

Comments documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search a comment with the content of "unit_test"	db.comments.find({"content": "unit_test"})	Return nothing	Return nothing	PASS	
2	insert a comment with the content of "unit_test"	db.comments.insertOne({"author_name": "COMP90000", "content": "unit_test"})	<pre>{ acknowledged: true, insertedId: ObjectId("608facfd5c58d53264afe58d") }</pre>	<pre>{ acknowledged: true, insertedId: ObjectId("608facfd5c58d53264afe58d") }</pre>	PASS	

3	search a comment with the content of "unit_test"	db.comments.find({"content": "unit_test"})	<pre>{ _id: ObjectId("608facfd5c58d53264afe58d"), author_name: 'COMP90000', content: 'unit_test' }</pre>	<pre>{ _id: ObjectId("608facfd5c58d53264afe58d"), author_name: 'COMP90000', content: 'unit_test' }</pre>	PASS	
4	update a comment's author_name from 'COMP90000' to 'COMP90001'	db.comments.updateOne({content: "unit_test"}, {\$set: {"author_name": "COMP90001"}}) db.comments.find({title: "unit_test"})	<pre>{ _id: ObjectId("608facfd5c58d53264afe58d"), author_name: 'COMP90001', content: 'unit_test' }</pre>	<pre>{ _id: ObjectId("608facfd5c58d53264afe58d"), author_name: 'COMP90001', content: 'unit_test' }</pre>	PASS	
5	delete the document of "unit_test"	db.comments.deleteOne({content: "unit_test"}) db.comments.find({content: "unit_test"})	Return nothing	Return nothing	PASS	
6	insert a new comment with an empty document	db.comments.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a new comment with author_name only	db.comments.insertOne({author_name: "unit_test"})	<pre>{ acknowledged: true, insertedId: ObjectId("608fad975c58d53264afe58e") }</pre>	Error message saying missing value for content	FAIL	should at least make the content as the mandatory fields

Articles documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
1	search an article with the content of "unit_test"	db.articles.find({"content": "unit_test"})	Return nothing	Return nothing	PASS	
2	insert an article with the content of "unit_test"	db.articles.insertOne({"title": "COMP90000", "content": "unit_test", "is_pending": "false"})	<pre>{ acknowledged: true, insertedId: ObjectId("608fb02c5c58d53264afe58f") }</pre>	<pre>{ acknowledged: true, insertedId: ObjectId("608fb02c5c58d53264afe58f") }</pre>	PASS	
3	search an article with the content of "unit_test"	db.articles.find({"content": "unit_test"})	<pre>{ _id: ObjectId("608fb02c5c58d53264afe58f"), title: 'COMP90000', content: 'unit_test', is_pending: 'false' }</pre>	<pre>{ _id: ObjectId("608fb02c5c58d53264afe58f"), title: 'COMP90000', content: 'unit_test', is_pending: 'false' }</pre>	PASS	

4	update an article's title from 'COMP90000' to 'COMP90001'	<pre>db.articles.updateOne({content: "unit_test"}, {\$set: {"title": "COMP90001"}}) db.articles.find({content: "unit_test"})</pre>	<pre>{ _id: ObjectId("608fb02c5c58d53264afe58f") , title: 'COMP90001', content: 'unit_test', is_pending: 'false' }</pre>	<pre>{ _id: ObjectId("608fb02c5c58d53264afe58f"), title: 'COMP90001', content: 'unit_test', is_pending: 'false' }</pre>	PASS	
5	delete the document of "unit_test"	<pre>db.articles.deleteOne({content: "unit_test"}) db.articles.find({content: "unit_test"})</pre>	Return nothing	Return nothing	PASS	
6	insert a new article with an empty document	db.articles.insertOne()	MongoshInvalidInputError: Missing required argument	MongoshInvalidInputError: Missing required argument	PASS	
7	insert a new article with author_id only	db.articles.insertOne({author_id: "unit_test"})	<pre>{ acknowledged: true, insertedId: ObjectId("608fb13f5c58d53264afe590") }</pre>	Error message saying missing value for title, content, author_id, and subjects	FAIL	should at least make the title, content, author_id, and subjects as the mandatory fields

Database Unit Test for sprint 2

This database unit test is to re-test the failed cases from [Database Unit Test for sprint 1](#) after adding the Database schema validation for each collection and make sure they will pass the test.

Please note: the ID is the same as the unit test in sprint 1.

Users documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a record only with a password	<pre>db.users.insertOne({password: "unit_test"}) db.users.find({password: "unit_test"})</pre>	MongoError: Document failed validation	MongoError: Document failed validation	PASS	
8	insert a user with an account only	<pre>db.users.insertOne({account: "unit_test"}) db.users.find({account: "unit_test"})</pre>	MongoError: Document failed validation	MongoError: Document failed validation	PASS	
9	insert a user with a student_number that exceeds 8 numbers	<pre>db.users.insertOne({account: "unit_test", password: "password", student_number: "123456789", is_moderator: false, is_admin: false}) db.users.find({account: "unit_test"})</pre>	MongoError: Document failed validation	MongoError: Document failed validation	PASS	
10	insert a user with the password of "testtest" when there is already a user with a password of "testtest"	<pre>db.users.insertOne({account: "testtest", password: "testtest", student_number: "2233445", is_moderator: "false", is_admin: "false"}) db.users.find({account: "testtest"}) db.users.insertOne({account: "testtesttest", password: "testtest", student_number: "1133445", is_moderator: "false", is_admin: "false"}) db.users.find({account: "testtesttest"})</pre>	<pre>{ _id: ObjectId("60b0f6b4f6c5093c706654d4"), account: 'testtest', password: 'testtest', student_number: '2233445', is_moderator: 'false', is_admin: 'false' } { _id: ObjectId("60b0f6e7f6c5093c706654d5"), account: 'testtesttest', password: 'testtest', student_number: '1133445', is_moderator: 'false', is_admin: 'false' }</pre>	<pre>{ _id: ObjectId("60b0f6b4f6c5093c706654d4"), account: 'testtest', password: 'testtest', student_number: '2233445', is_moderator: 'false', is_admin: 'false' } { _id: ObjectId("60b0f6e7f6c5093c706654d5"), account: 'testtesttest', password: 'testtest', student_number: '1133445', is_moderator: 'false', is_admin: 'false' }</pre>	PASS	

Tools documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a tool with articles only	db.tools.insertOne({articles: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Subjects documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a subject with a name but an empty value for subject_code and description	db.subjects.insertOne({name: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	
8	insert a subject with a description but an empty value for subject_code and name	db.subjects.insertOne({description: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Sections documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a section with the comments only	db.sections.insertOne({comments: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Pendingarticles documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a pending article with the content only	db.pendingarticles.insertOne({content: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Comments documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a new comment with author_name only	db.comments.insertOne({author_name: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Articles documents						
ID	Action	Input	Actual output	Expected output	Test result	Comments
7	insert a new article with author_id only	db.articles.insertOne({author_id: "unit_test"})	MongoError: Document failed validation	MongoError: Document failed validation	PASS	

Integration test for Sprint 1

CI/CD		
API	Result	Output
Backend Status: http://api.cervidae.com.au:8000/rest/info	PASS	<pre>rsrepo.service - RS Repo Backend Loaded: loaded (/etc/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 06:20:18 UTC; 24min ago Main PID: 368758 (node) Tasks: 11 (limit: 1110) Memory: 63.5M CGroup: /system.slice/rsrepo.service 368758 /usr/bin/node index.js Apr 30 06:20:18 instance-20210331-1946 systemd[1]: Started RS Repo Backend. Apr 30 06:20:21 instance-20210331-1946 node[368758]: Warning: connect.session() MemoryStore is not Apr 30 06:20:21 instance-20210331-1946 node[368758]: designed for a production environment, as it will leak Apr 30 06:20:21 instance-20210331-1946 node[368758]: memory, and will not scale past a single process. Apr 30 06:20:21 instance-20210331-1946 node[368758]: Listening on port 4000 Apr 30 06:20:23 instance-20210331-1946 node[368758]: Connected to mongodb+srv://root:root@cluster0.uhlwx.mongodb.net/myFirstDatabase?retryWrites=true&w=majority.. Apr 30 06:20:23 instance-20210331-1946 node[368758]: existed the root account, no need to create</pre>
Backend Rebuild: http://api.cervidae.com.au:8000/rest/rebuild	PASS	<pre>Rebuild succeeded. Restart succeeded. rsrepo.service - RS Repo Backend Loaded: loaded (/etc/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 06:44:44 UTC; 51ms ago Main PID: 369199 (node) Tasks: 6 (limit: 1110) Memory: 2.3M CGroup: /system.slice/rsrepo.service 369199 /usr/bin/node index.js Apr 30 06:44:44 instance-20210331-1946 systemd[1]: Started RS Repo Backend.</pre>
Backend Restart: http://api.cervidae.com.au:8000/rest/restart	PASS	<pre>Restart succeeded. rsrepo.service - RS Repo Backend Loaded: loaded (/etc/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 06:44:45 UTC; 18ms ago Main PID: 369222 (node) Tasks: 2 (limit: 1110) Memory: 1.5M CGroup: /system.slice/rsrepo.service 369222 /usr/bin/node index.js Apr 30 06:44:45 instance-20210331-1946 systemd[1]: Started RS Repo Backend.</pre>
Quick-Check: http://api.cervidae.com.au:8000/rest/check (0:inactive, 1:active)	PASS	1

One-click Deployment

API	Result	Output
one-click-setup.sh	PASS	<pre>one-click-setup.sh Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ... Connecting to raw.githubusercontent.com (raw.githubusercontent.com) 185.199.108.133 :443... connected. HTTP request sent, awaiting response... 200 OK Length: 943 [text/plain] Saving to: 'one-click-setup.sh' one-click-setup.sh 100%[=====] 943 --.-KB/s in 0s 2021-04-30 11:30:52 (66.2 MB/s) - 'one-click-setup.sh' saved [943/943] Configuring firewall. Rules updated Rules updated (v6) Rules updated Rules updated (v6) Rules updated Rules updated (v6) A. Cloning from Github Source. Cloning into 'COMP90082-RS-2021'... remote: Enumerating objects: 993, done. remote: Counting objects: 100% (993/993), done. remote: Compressing objects: 100% (544/544), done. remote: Total 993 (delta 607), reused 787 (delta 413), pack-reused 0 Receiving objects: 100% (993/993), 25.21 MiB 4.95 MiB/s, done. Resolving deltas: 100% (607/607), done. Branch 'backend/develop' set up to track remote branch 'backend/develop' from 'origin'. Switched to a new branch 'backend/develop' 3. Install systemd service Done! Done! ● rsadmin.service - RS Server Admin API Loaded: loaded (/lib/systemd/system/rsadmin.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 11:34:32 UTC; 1min 14s ago Main PID: 12569 (python3) Tasks: 3 (limit: 1107) Memory: 56.0M CGroup: /system.slice/rsadmin.service └─12569 /usr/bin/python3 manage.py runserver 0:8000 └─12645 /usr/bin/python3 manage.py runserver 0:8000 Apr 30 11:34:32 instance-20210430-1928 systemd[1]: Started RS Server Admin API. Apr 30 11:34:34 instance-20210430-1928 python3[12645]: Watching for file changes with StatReloader Apr 30 11:34:34 instance-20210430-1928 python3[12645]: Performing system checks... Apr 30 11:34:34 instance-20210430-1928 python3[12645]: System check identified no issues (0 silenced). Apr 30 11:34:34 instance-20210430-1928 python3[12645]: April 30, 2021 - 11:34:34 Apr 30 11:34:34 instance-20210430-1928 python3[12645]: Django version 3.2, using settings 'api.settings' Apr 30 11:34:34 instance-20210430-1928 python3[12645]: Starting development server at http://0:8000/ Apr 30 11:34:34 instance-20210430-1928 python3[12645]: Quit the server with CONTROL-C. ● rsrepo.service - RS Repo Backend Loaded: loaded (/lib/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 11:35:46 UTC; 304ms ago Main PID: 12943 (node) Tasks: 6 (limit: 1107) Memory: 3.7M CGroup: /system.slice/rsrepo.service └─12943 /usr/bin/node index.js Apr 30 11:35:46 instance-20210430-1928 systemd[1]: Started RS Repo Backend. ubuntu@instance-20210430-1928:~\$</pre>

Server		
Function	Result	Output
connection	PASS	<pre>when back end has errors: rsrepo.service - RS Repo Backend Loaded: loaded (/etc/systemd/system/rsrepo.service; disabled; vendor preset: enabled) Active: active (running) since Fri 2021-04-30 11:45:51 UTC; 6min ago Main PID: 373304 (node) Tasks: 11 (limit: 1110) Memory: 71.8M CGroup: /system.slice/rsrepo.service 373304 /usr/bin/node index.js Apr 30 11:49:03 instance-20210331-1946 node[373304]: at ObjectId.SchemaType. applySetters (/var/node/backend/node_modules/mongoose/lib/schematype.js:1104:12) Apr 30 11:49:03 instance-20210331-1946 node[373304]: at model.\$set (/var/node /backend/node_modules/mongoose/lib/document.js:1279:20) Apr 30 11:49:03 instance-20210331-1946 node[373304]: at model.\$set (/var/node /backend/node_modules/mongoose/lib/document.js:1023:16) Apr 30 11:49:03 instance-20210331-1946 node[373304]: at model.Document (/var /node/backend/node_modules/mongoose/lib/document.js:148:12) Apr 30 11:49:03 instance-20210331-1946 node[373304]: at model.Model (/var/node /backend/node_modules/mongoose/lib/model.js:105:12) Apr 30 11:49:03 instance-20210331-1946 node[373304]: at new model (/var/node</pre>


```
/backend/node_modules/mongoose/lib/model.js:4713:15)
Apr 30 11:49:03 instance-20210331-1946 node[373304]: newAndSave (/var/node/backend/proxies/comment.js:41:22) at Object.exports.
Apr 30 11:49:03 instance-20210331-1946 node[373304]: /node/backend/controllers/comment.js:9:13) at exports.NewAndSave (/var
Apr 30 11:49:03 instance-20210331-1946 node[373304]: handle_request] (/var/node/backend/node_modules/express/lib/router/layer.js:95:5) at Layer.handle [as
Apr 30 11:49:03 instance-20210331-1946 node[373304]: /node_modules/express/lib/router/route.js:137:13) at next (/var/node/backend
Apr 30 11:49:03 instance-20210331-1946 node[373304]: /node/backend/middlewares/auth.js:19:9) at module.exports (/var
Apr 30 11:49:03 instance-20210331-1946 node[373304]: (internal/process/next_tick.js:68:7) } }, at process._tickCallback
Apr 30 11:49:03 instance-20210331-1946 node[373304]: } _message: 'Comment validation failed'
Apr 30 11:50:01 instance-20210331-1946 node[373304]: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJfawQioiI2MDUzMmU4ZDdiYTM1MTRiNTc2ZWJiMjMiLCJfYWRtaW4iOmZhbHN1LlCjPjY
XQioje2MTk3ODMxMDh9.U4u-3Bj4JDFw5rnpbOjY3YcZtlhF1F4ollDILhpQv0A
Apr 30 11:50:01 instance-20210331-1946 node[373304]: verifying token
Apr 30 11:50:01 instance-20210331-1946 node[373304]: { _id: '60532e8d7ba3514b576ebb23',
Apr 30 11:50:01 instance-20210331-1946 node[373304]: _moderator: true,
Apr 30 11:50:01 instance-20210331-1946 node[373304]: _admin: false,
Apr 30 11:50:01 instance-20210331-1946 node[373304]: iat: 1619783108 }
Apr 30 11:50:55 instance-20210331-1946 node[373304]: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJfawQioiI2MDUzMmU4ZDdiYTM1MTRiNTc2ZWJiMjMiLCJfYWRtaW4iOmZhbHN1LlCjPjY
XQioje2MTk3ODMxMDh9.U4u-3Bj4JDFw5rnpbOjY3YcZtlhF1F4ollDILhpQv0A
Apr 30 11:50:55 instance-20210331-1946 node[373304]: verifying token
Apr 30 11:50:55 instance-20210331-1946 node[373304]: { _id: '60532e8d7ba3514b576ebb23',
Apr 30 11:50:55 instance-20210331-1946 node[373304]: _moderator: true,
Apr 30 11:50:55 instance-20210331-1946 node[373304]: _admin: false,
Apr 30 11:50:55 instance-20210331-1946 node[373304]: iat: 1619783108 }
Apr 30 11:51:34 instance-20210331-1946 node[373304]: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJfawQioiI2MDUzMmU4ZDdiYTM1MTRiNTc2ZWJiMjMiLCJfYWRtaW4iOmZhbHN1LlCjPjY
XQioje2MTk3ODMxMDh9.U4u-3Bj4JDFw5rnpbOjY3YcZtlhF1F4ollDILhpQv0A
Apr 30 11:51:34 instance-20210331-1946 node[373304]: verifying token
Apr 30 11:51:34 instance-20210331-1946 node[373304]: { _id: '60532e8d7ba3514b576ebb23',
Apr 30 11:51:34 instance-20210331-1946 node[373304]: _moderator: true,
Apr 30 11:51:34 instance-20210331-1946 node[373304]: _admin: false,
Apr 30 11:51:34 instance-20210331-1946 node[373304]: iat: 1619783108 }
Apr 30 11:51:42 instance-20210331-1946 node[373304]: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJfawQioiI2MDUzMmU4ZDdiYTM1MTRiNTc2ZWJiMjMiLCJfYWRtaW4iOmZhbHN1LlCjPjY
XQioje2MTk3ODMxMDh9.U4u-3Bj4JDFw5rnpbOjY3YcZtlhF1F4ollDILhpQv0A
Apr 30 11:51:42 instance-20210331-1946 node[373304]: verifying token
Apr 30 11:51:42 instance-20210331-1946 node[373304]: { _id: '60532e8d7ba3514b576ebb23',
Apr 30 11:51:42 instance-20210331-1946 node[373304]: _moderator: true,
Apr 30 11:51:42 instance-20210331-1946 node[373304]: _admin: false,
Apr 30 11:51:42 instance-20210331-1946 node[373304]: iat: 1619783108 }
Apr 30 11:51:46 instance-20210331-1946 node[373304]: new article submitted
Apr 30 11:51:46 instance-20210331-1946 node[373304]: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJfawQioiI2MDUzMmU4ZDdiYTM1MTRiNTc2ZWJiMjMiLCJfYWRtaW4iOmZhbHN1LlCjPjY
XQioje2MTk3ODMxMDh9.U4u-3Bj4JDFw5rnpbOjY3YcZtlhF1F4ollDILhpQv0A
Apr 30 11:51:46 instance-20210331-1946 node[373304]: verifying token
Apr 30 11:51:46 instance-20210331-1946 node[373304]: { _id: '60532e8d7ba3514b576ebb23',
Apr 30 11:51:46 instance-20210331-1946 node[373304]: _moderator: true,
Apr 30 11:51:46 instance-20210331-1946 node[373304]: _admin: false,
Apr 30 11:51:46 instance-20210331-1946 node[373304]: iat: 1619783108 }
Apr 30 11:51:51 instance-20210331-1946 node[373304]: { Error: Comment validation failed:
section_id: Cast to ObjectId failed for value "" at path "section_id"
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at ValidationError.inspect (/var
/node/backend/node_modules/mongoose/lib/error/validation.js:47:26)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at formatValue (internal/util
/inspect.js:491:31)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at inspect (internal/util/inspect.
js:189:10)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Object.formatWithOptions (util.
js:84:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Console.(anonymous function)
(console.js:191:15)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Console.log (console.js:202:31)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Comment.newAndSave (/var/node
/backend/controllers/comment.js:11:21)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at /var/node/backend/node_modules
/mongoose/lib/model.js:4870:16
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at /var/node/backend/node_modules
/mongoose/lib/helpers/promiseOrCallback.js:16:11
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at /var/node/backend/node_modules
/mongoose/lib/model.js:4893:21
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at $__save.error (/var/node/backend
/node_modules/mongoose/lib/model.js:500:16)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at /var/node/backend/node_modules
/kareem/index.js:247:48
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at next (/var/node/backend
/node_modules/kareem/index.js:168:27)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at next (/var/node/backend
```

```

/node_modules/kareem/index.js:170:9)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at next (/var/node/backend
/node_modules/kareem/index.js:170:9)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Kareem.execPost (/var/node
/backend/node_modules/kareem/index.js:218:3)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: errors:
Apr 30 11:51:51 instance-20210331-1946 node[373304]: { section_id:
Apr 30 11:51:51 instance-20210331-1946 node[373304]: { CastError: Cast to ObjectId
failed for value "" at path "section_id"
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at ObjectId.cast (/var/node
/backend/node_modules/mongoose/lib/schema/objectid.js:281:11)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at ObjectId.SchemaType.
applySetters (/var/node/backend/node_modules/mongoose/lib/schematype.js:1104:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.$set (/var/node
/backend/node_modules/mongoose/lib/document.js:1279:20)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.$set (/var/node
/backend/node_modules/mongoose/lib/document.js:1023:16)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.Document (/var/node
/backend/node_modules/mongoose/lib/document.js:148:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.Model (/var/node
/backend/node_modules/mongoose/lib/model.js:105:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at new model (/var/node
/backend/node_modules/mongoose/lib/model.js:4713:15)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Object.exports.newAndSave (
/var/node/backend/proxies/comment.js:41:22)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at exports.NewAndSave (/var
/node/backend/controllers/comment.js:9:13)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Layer.handle [as
handle_request] (/var/node/backend/node_modules/express/lib/router/layer.js:95:5)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at next (/var/node/backend
/node_modules/express/lib/router/route.js:137:13)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at module.exports (/var/node
/backend/middlewares/auth.js:19:9)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at process._tickCallback
(internal/process/next_tick.js:68:7)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: stringValue: '',
Apr 30 11:51:51 instance-20210331-1946 node[373304]: messageFormat: undefined,
Apr 30 11:51:51 instance-20210331-1946 node[373304]: kind: 'ObjectId',
Apr 30 11:51:51 instance-20210331-1946 node[373304]: value: '',
Apr 30 11:51:51 instance-20210331-1946 node[373304]: path: 'section_id',
Apr 30 11:51:51 instance-20210331-1946 node[373304]: reason:
Apr 30 11:51:51 instance-20210331-1946 node[373304]: Error: Argument passed in must
be a single String of 12 bytes or a string of 24 hex characters
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at new ObjectId (/var/node
/backend/node_modules/bson/lib/bson/objectid.js:59:11)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at castObjectId (/var/node
/backend/node_modules/mongoose/lib/cast/objectid.js:25:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at ObjectId.cast (/var/node
/backend/node_modules/mongoose/lib/schema/objectid.js:279:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at ObjectId.SchemaType.
applySetters (/var/node/backend/node_modules/mongoose/lib/schematype.js:1104:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.$set (/var/node
/backend/node_modules/mongoose/lib/document.js:1279:20)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.$set (/var/node
/backend/node_modules/mongoose/lib/document.js:1023:16)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.Document (/var
/node/backend/node_modules/mongoose/lib/document.js:148:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at model.Model (/var/node
/backend/node_modules/mongoose/lib/model.js:105:12)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at new model (/var/node
/backend/node_modules/mongoose/lib/model.js:4713:15)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Object.exports.
newAndSave (/var/node/backend/proxies/comment.js:41:22)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at exports.NewAndSave (/var
/node/backend/controllers/comment.js:9:13)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at Layer.handle [as
handle_request] (/var/node/backend/node_modules/express/lib/router/layer.js:95:5)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at next (/var/node/backend
/node_modules/express/lib/router/route.js:137:13)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at module.exports (/var
/node/backend/middlewares/auth.js:19:9)
Apr 30 11:51:51 instance-20210331-1946 node[373304]: at process._tickCallback
(internal/process/next_tick.js:68:7) } },
Apr 30 11:51:51 instance-20210331-1946 node[373304]: _message: 'Comment validation failed'
}

```

del
ay

PASS

```
[→ ~ ping 158.101.158.250
PING 158.101.158.250 (158.101.158.250): 56 data bytes
64 bytes from 158.101.158.250: icmp_seq=0 ttl=53 time=149.794 ms
64 bytes from 158.101.158.250: icmp_seq=1 ttl=53 time=153.131 ms
64 bytes from 158.101.158.250: icmp_seq=2 ttl=53 time=140.253 ms
64 bytes from 158.101.158.250: icmp_seq=3 ttl=53 time=145.626 ms
64 bytes from 158.101.158.250: icmp_seq=4 ttl=53 time=148.720 ms
^C
--- 158.101.158.250 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 140.253/147.505/153.131/4.347 ms
→ ~ █
```

Integration test for Sprint 2

auto-deploy on server		
nginx service	PASS	<pre>● nginx.service - A high performance web server and a reverse proxy server Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled) Active: active (running) since Wed 2021-05-19 14:52:45 UTC; 57ms ago Docs: man:nginx(8) Process: 6552 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS) Process: 6569 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS) Main PID: 6575 (nginx) Tasks: 3 (limit: 4714) Memory: 4.8M CGroup: /system.slice/nginx.service └─6575 nginx: master process /usr/sbin/nginx -g daemon on; master_process on; └─6576 nginx: worker process └─6577 nginx: worker process</pre>
lockfile saving	PASS	<pre>[4/4] Building fresh packages... success Saved lockfile. Done in 247.50s. yarn run v1.22.10 \$ react-scripts build Creating an optimized production build...</pre>

Server								
performance	<div>PASS</div>		total	used	used	shared	buff/cache	available
		Mem:	976Mi	283Mi	185Mi	0.0Ki	507Mi	564Mi
		Swap:	2.0Gi	97Mi	1.9Gi	-	-	-

Schema validation test

Based on [DB schema validation rules](#), schema validation has been tested on the samples listed below for each collection.

Users

ID	Description	Error message	Incorrect example
1.	if missing one of 'account', 'student_number', 'password', 'is_moderator', 'is_admin' value	" MongoError : Document failed validation"	db.users.insertOne({account: "unit_test"})
2.	if the length of a student_number is not between 6 and 8	" MongoError : Document failed validation"	db.users.insertOne({account: "testtest", password: "testtest", student_number: "223344509", is_moderator: "false", is_admin: "false"})
3.	if the length of a password is not at least 6 digits	" MongoError : Document failed validation"	db.users.insertOne({account: "testtest", password: "test", student_number: "2233447", is_moderator: "false", is_admin: "false"})
4.	if the value of student_number or account is not unique	" MongoError : Document failed validation"	db.users.insertOne({account: "testtest", password: "testtest", student_number: "2233447", is_moderator: "false", is_admin: "false"}) db.users.insertOne({account: "testtest", password: "test21", student_number: "2233441", is_moderator: "false", is_admin: "false"}) or db.users.insertOne({account: "tester", password: "testtest", student_number: "2233447", is_moderator: "false", is_admin: "false"})

Tools

ID	Description	Error message	Incorrect example
1.	if missing value of 'name'	" MongoError : Document failed validation"	db.tools.insertOne({articles: "unit_test"})
2.	if the value of name is not unique	" MongoError : Document failed validation"	db.tools.insertOne({name: "unit_test"}) db.tools.insertOne({name: "unit_test"})

Subjects

ID	Description	Error message	Incorrect example
1.	if missing one of 'name', 'subject_code', 'created_by', 'description' value	" MongoError : Document failed validation"	db.subjects.insertOne({name: "unit_test"})
2.	if the value of name or subject_code is not unique	" MongoError : Document failed validation"	db.subjects.insertOne({name: "unit_test", subject_code: "COMP90082", created_by: "tester", description: "just test"}) db.subjects.insertOne({name: "unit_test", subject_code: "COMP90013", created_by: "tester", description: "just test"}) or db.subjects.insertOne({name: "unit", subject_code: "COMP90082", created_by: "tester", description: "just test"})

Sections

ID	Description	Error message	Incorrect example
1.	if missing one of 'name', 'owner', 'type', 'articles' value	" MongoError : Document failed validation"	db.sections.insertOne({comments: "unit_test"})

Pending articles

ID	Description	Error message	Incorrect example
1.	if missing one of 'subjects', 'title', 'editor_id', 'content' value	" MongoError : Document failed validation"	<code>db.pendingarticles.insertOne({content:"unit_test"})</code>

Comments

ID	Description	Error message	Incorrect example
1.	if missing one of 'content', 'author_id', 'root_comment_id', 'section_id', 'author_name' value	" MongoError : Document failed validation"	<code>db.comments.insertOne({author_name:"unit_test"})</code>

Articles

ID	Description	Error message	Incorrect example
1.	if missing one of 'content', 'title', 'author_id', 'subjects', 'is_pending' value	" MongoError : Document failed validation"	<code>db.articles.insertOne({author_id: "unit_test"})</code>
2.	if the value of name or title is not unique	" MongoError : Document failed validation"	<code>db.articles.insertOne({author_id: "unit_test", content:"unit_test", title:"just check", is_pending: "false"})</code> <code>db.articles.insertOne({author_id: "unit", content:"just test", title:"just check", is_pending: "false"})</code>