**Royal Melbourne Institute of Technology**

COSC 2299 - Software Engineering Process & Tools

**Online Booking System**

**Test Cases**

Version 1.0

**Members:**

SangYeon Lee - s3742751

Mitchell Gust - s3782095

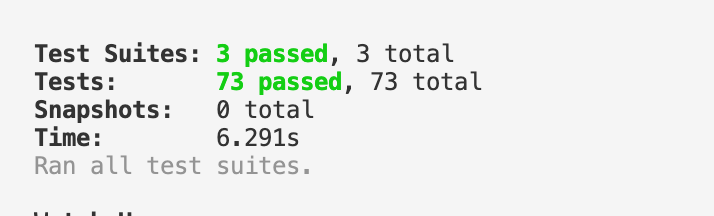
David Brown - s3785994

**Intro**

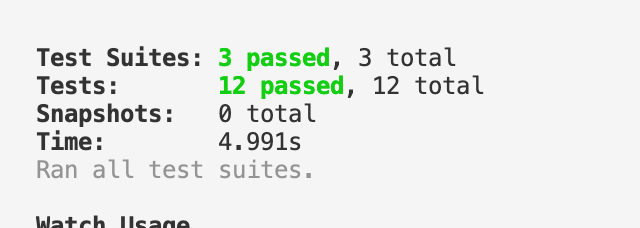
The following document lists all test cases used within the backend of the system implementation, as well as evidence of their execution on both local machines and CircleCI.

**Evidence (local machine)**

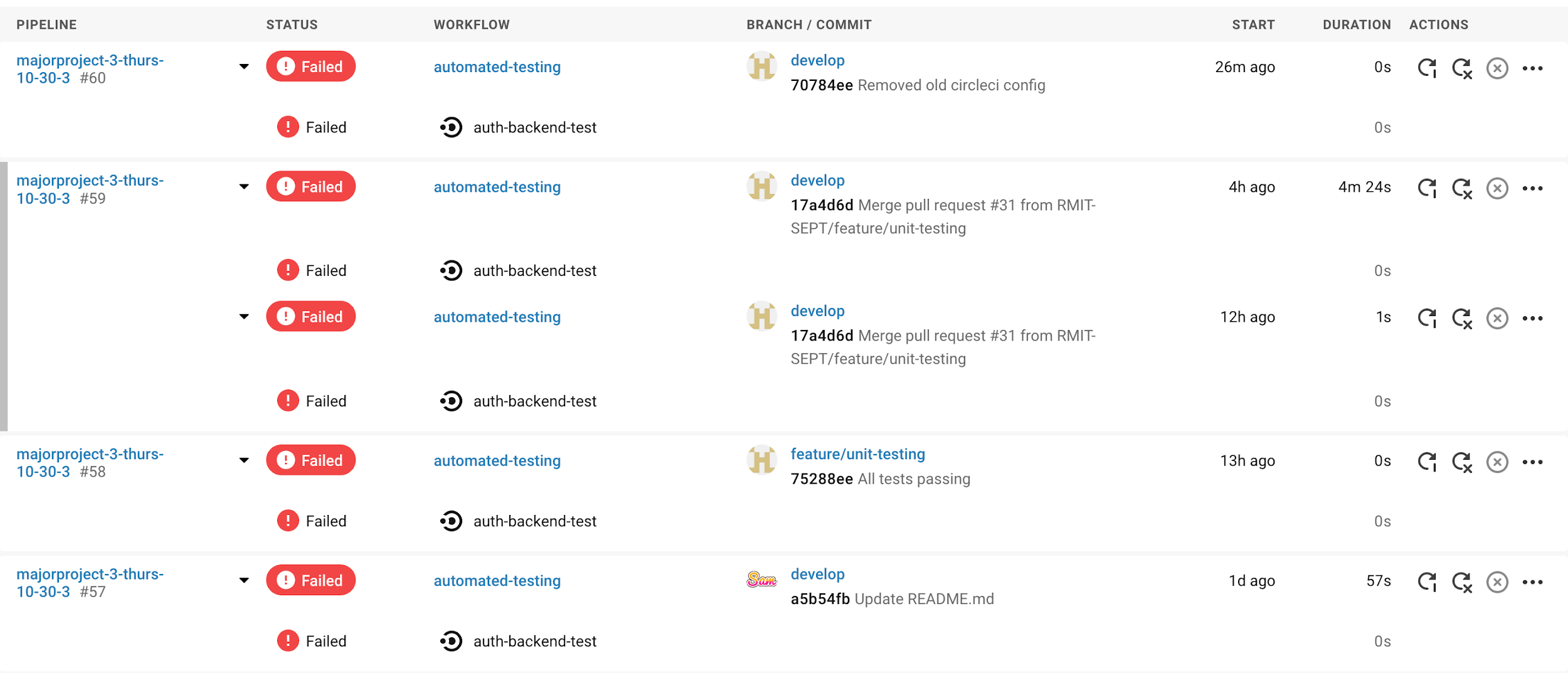
**Auth microservice (‘auth’ directory)**



**Business microservice (‘business’ directory)**

****

**Evidence (CircleCI)**



**Test Case Structure And Details**

Test cases are located within ‘\_\_test\_\_’ directories located within individual microservices, divided into files depending on the category of functionality tested (e.g. ‘signout.test.ts’ tests signing out functionality). Tests run using the SuperTest library, which performs API calls on the routes provided by the ExpressJS REST APIs produced by the microservices. Jest is used to hold and run tests using the ‘npm test’ command.

An issue was encountered in some cases when one microservice would use the other microservice in some form (e.g. where created employees were needed for some business activities). In these cases, it was impossible to solve the problems given the restrictions (e.g. inability to change implementation without ruining purpose of unit tests, small amount of time due to loss of team members and sudden change in language and framework). Therefore, for the time being, these tests have been removed.

**Tests**

Relevant tests conducted for each user type.

**Sign In (signin.test.ts)**

* Tests requests containing missing fields
  + Username, password
* Tests failure due to incorrect password
* Tests successful login

**Sign Up (signup.test.ts)**

Relevant tests conducted for each user type.

* Tests requests containing missing fields
  + Username, password, name, address, phone, secret (super admin), businessId (admin, worker), days (worker), shift (worker)
* Tests failure due to existing user with same email/username
* Tests failure to to incorrect permissions
  + Secret key for super creation is incorrect
  + Not logged in or providing cookie required for creation of admin accounts (super admin needed)
  + Not logged in or providing cookie required for creation of worker accounts (any admin account type needed)
* Tests successful signup and return of cookie

**Sign Out (signout.test.ts)**

* Tests successful sign out for each user type.

Note: since the sign out functionality just resets the session token server side for the current machine when called by the API, it ultimately does not require much negative testing

**Create Business (create-business.test.ts)**

* Tests attempting to create business with missing fields
  + Name, provided services, service type
* Tests attempting to create business without permission (super admin)
* Tests successful creation of businesses.

**Create Schedule (create-schedule.test.ts)**

* Tests failure in creating schedules for non-existing businesses.
* Tests failure in creating schedules for businesses with incorrectly supplied worker and/or business IDs.

**Get Business (get-business.test.ts)**

* Tests success in retrieving lists of all businesses attached to the system.
* Tests failure in retrieving business with ID that does not exist
* Tests success in retrieving existing business