SC2006-Software Engineering

Lab 4 Deliverables

Group Member	**Matric Number**
Lim Kiat Yang Ryan	U2421937D
Yu Wenhao	U2421425F
Yong Chee Seng	U2420563K
Peng Sizhe	U2423895H
Tarun Ilangovan	U2422251A

A. Frontend Testing

A.1 Control Class: AuthController

Method	Description
login(loginRequest: LoginRequest)	Passes a loginRequest object to AuthService after user attemps to login
signup(signupRequest:SignupRequest)	Passes a signupRequest object to AuthService after user signs up

The AuthController class is an important class that handles login and signup requests whenever a user tries to log in to an existing account or sign up for a new account. It implements most of the description of requirement 5 in our requirements elicitation.

5. The system shall enforce authentication for all functions that require user identity
    5.1. The user shall be able to create an account.
        5.1.1. The user shall provide a username.
            5.1.1.1. The username shall be a string of 6-14 characters.
            5.1.1.2. The username shall be unique across the user database.
        5.1.2. The user shall provide a password.
            5.1.2.1. The password shall be a string of at least 8 characters.
            5.1.2.2. The password shall meet minimum complexity requirements:
                5.1.2.2.1. The password shall have at least 1 lowercase letter.
                5.1.2.2.2. The password shall have at least 1 uppercase alphabet.
                5.1.2.2.3. The password shall have at least 1 number.
                5.1.2.2.4. The password shall have at least 1 non-alphanumeric character.
    5.2. The user shall be able to log into their account.
        5.2.1. The user shall be able to log into their account with valid credentials (username and password).
    5.3. The system shall save user sessions using a secure method after successful login.
        5.3.1. The system shall automatically invalidate user sessions no later than 7 days after the last successful login.

AuthController has two functions.

Sign Up:
This function is available when a user does not have an existing account, or wants to create a new account. He/She will be redirected here from either (1) the home page by clicking the ”Sign Up” button, (2) the log-in page by clicking the “Don’t have an account?” button. He/She will be prompted to enter a username and a password.

If the username is not unique or not between 6-14 characters, it will be rejected and the user will be prompted to enter a valid username.

If the password is not at least 8 characters, and does not meet the minimum complexity requirements as stated above, it will be rejected and the user will be prompted to enter a valid password.

If both username and password are valid, the account will be created and the user will automatically be logged in, with all the functions available to him/her.

Login:
This function is available when a user has an existing account and he/she tries to log in to that account from the log in page. The user will be navigated to the login page from the home page by selecting the login button. He/She will be prompted to enter a username and a password corresponding to the username. If valid credentials are given, the user will be logged in and can have access to authenticated functions.

A.2 Equivalent Class Testing

We identified the following equivalence classes separately for the login and sign-up functions.

A.2.1 Sign Up

Category	Input Condition	Example Input	Validity	Expected Output	Intended Output
Username	Username has <6 characters	“user1”	Invalid	“Username should be 6-14 characters long” Error	“Username should be 6-14 characters long” Error
	Username has >14 characters	“username123345678”	Invalid	“Username should be 6-14 characters long” Error	“Username should be 6-14 characters long” Error
	Username already found in the database	“existinguser1”	Invalid	“User Already Exists” Error	“User Already Exists” Error
	Username has 6-14 characters and is unique	“uniqueuser1”	Valid	Successful	Successful
Password	Password has <8 characters	“passwor”	Invalid	“Password does not meet complexity requirements” Error	“Password does not meet complexity requirements” Error
	Password does not contain lowercase letters	“PASSWORD”	Invalid	“Password does not meet complexity requirements” Error	“Password does not meet complexity requirements” Error
	Password does not contain uppercase letters	“password”	Invalid	“Password does not meet complexity requirements” Error	“Password does not meet complexity requirements” Error
	Password does not contain numbers.	“Password”	Invalid	“Password does not meet	“Password does not meet

				complexity requirements" Error	complexity requirements" Error
	Password does not contain non-alphanumeric characters.	"Password123"	Invalid	"Password does not meet complexity requirements" Error	"Password does not meet complexity requirements" Error
	Password has 8 characters and meets minimum complexity requirements	"Password123%%"	Valid	Successful	Successful

A.2.2 Login

Input Condition	**Example Input**	**Validity**	**Expected Output**	**Intended Output**
Username is blank	Username: "" Password: "Password123%%"	Invalid	"Invalid Credentials" Error	"Invalid Credentials" Error
Password is blank	Username: "uniqueuser1" Password: ""	Invalid	"Invalid Credentials" Error	"Invalid Credentials" Error
Username is not in database	Username: "newuser1" Password: "Password123%%"	Invalid	"Invalid Credentials" Error	"Invalid Credentials" Error
Username and Password do not match in database	Username: "uniqueuser1" Password: "Password123%"	Invalid	"Invalid Credentials" Error	"Invalid Credentials" Error
Username and Password match in database	Username: "uniqueuser1" Password: "Password123%%"	Valid	Successful	Successful

A.3 Boundary Value Testing

As boundary value testing only works for value ranges, we will only be boundary value testing on the lengths of password and username during the signup function.

Boundary	Just Below	On Boundary	Just Above	Intended Output	Actual Output
Username Length - Lower Boundary (6)	5	6	7	Invalid/ Valid/ Valid	Invalid/ Valid/ Valid
Username Length - Higher Boundary (14)	13	14	15	Valid/ Valid/ Invalid	Valid/ Valid/ Invalid
Password Length - Lower Boundary (8)	7	8	9	Invalid/ Valid/ Valid	Invalid/ Valid/ Valid

B. Backend Testing

The backend includes automated tests to verify that all components function correctly before deployment.

Testing Framework: JUnit
Directory: /backend/edufinder/src/test/java/com/sc2006/g5/edufinder/

Unit Testing:

- **Approach**: White-box testing
- **Purpose**: To test each component in isolation by mocking its dependencies.
- **Scope**: Covers core services, controllers, and other critical classes to ensure each behaves as expected under various conditions.

B.1 Unit Testing

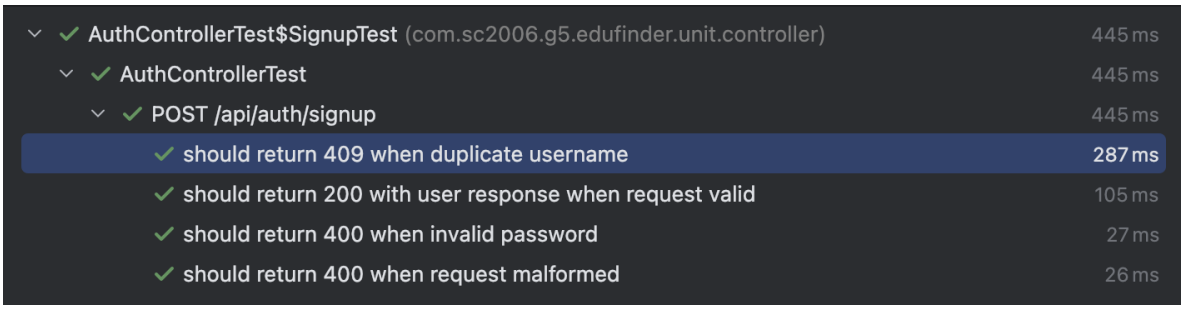
We chose to demonstrate unit testing using **AuthServiceTest** and **AuthControllerTest**, as these components contain more complex logic compared to others. The complete unit test results can be found in **/docs/Unit Test Results.html**.

B.1.1 Signup

Component: AuthController
Function:
Validates request input (username and password) before delegating to AuthService, and sets the authentication token issued by the service in the response cookie.

Test ID	Input	Equivalence Class	Expected Output
T1	username: valid_username password: AbCd123@	Valid	Status: 200 Token: Issued and Set Body: User Information
T2	{}	Malformed input	Status: 400
T3	password: AbCd123@	Missing Username	Status: 400
T4	username: valid_username	Missing Password	Status: 400
T5	username: valid_username password: ABCD123@	No Lower Case Password	Status: 400
T6	username: valid_username password: abcd123@	No Upper Case Password	Status: 400
T7	username: valid_username password: Abcdefg@	No Digit Password	Status: 400
T8	username: valid_username password: Abcd1234	No Special Symbol Password	Status: 400
T9	username: valid_username password: Abab12@	Short Password	Status: 400
T10	username: invalid_username password: AbCd123@	Service throw DuplicateUsernameException	Status: 409

Result:



Note that some test cases are grouped together due to similar nature:

1. T2, T3, T4
2. T5, T6, T7, T8, T9

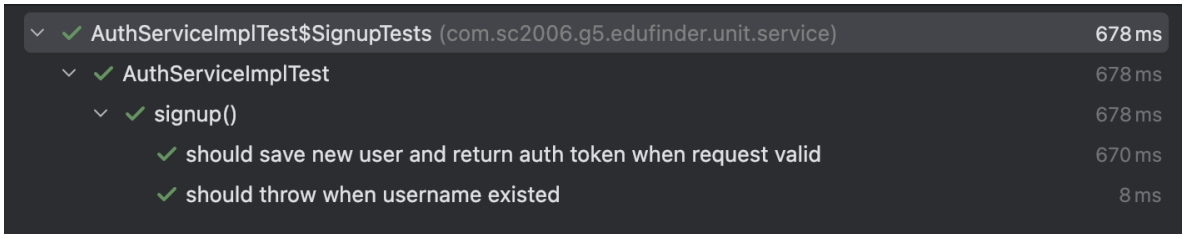
Component: AuthService

Function:

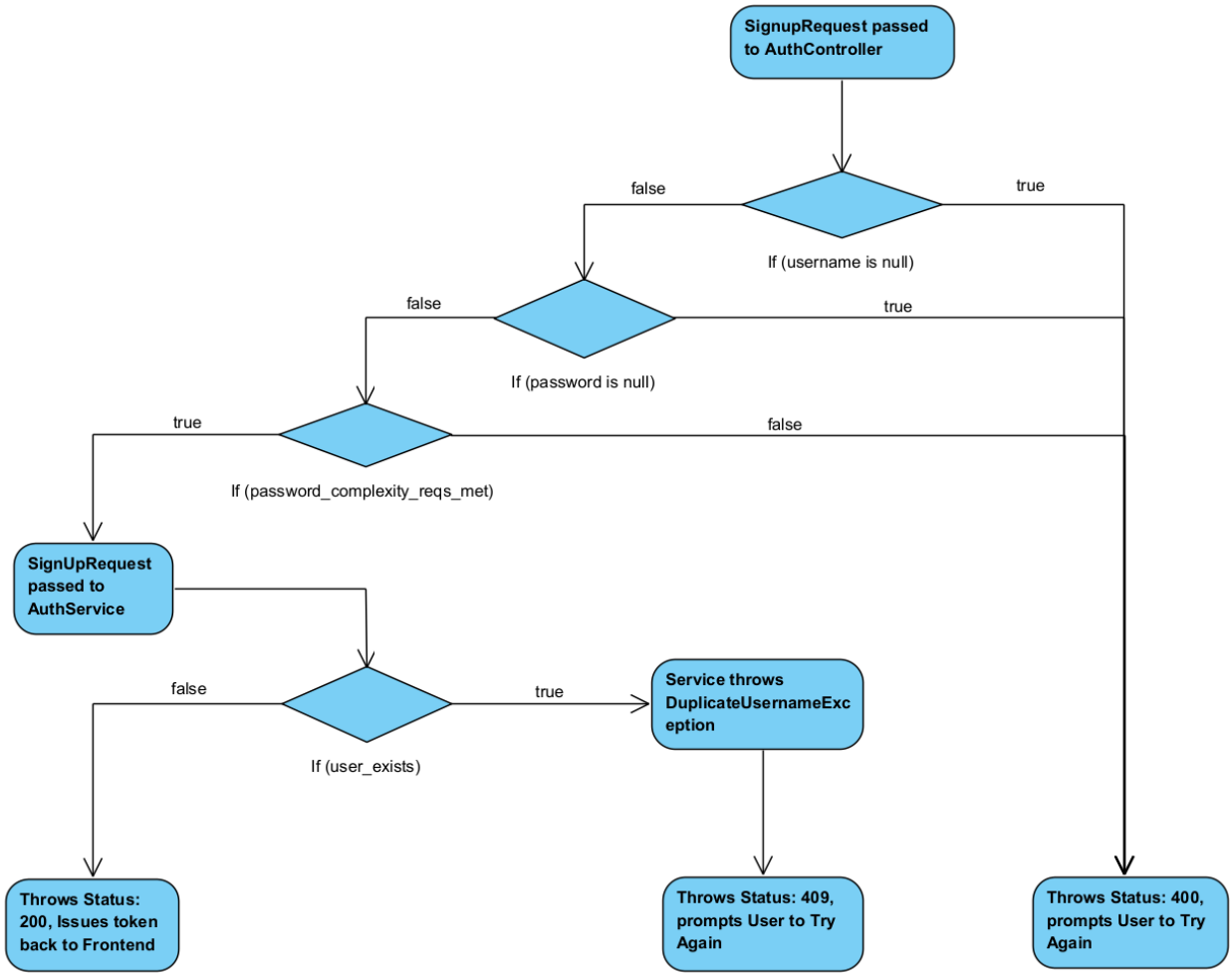
Check if the username exists, create a user and return a token.

Path ID	Predicate Coverage	Input Condition	Expected Output
P1	DuplicateUsername = false	Username does not exist	Password is encoded; user is saved; token with user ID is returned
P2	DuplicateUsername = true	Username already exists	DuplicatedUsernameException is thrown

Result:



Control Flow Diagram (For Signup)

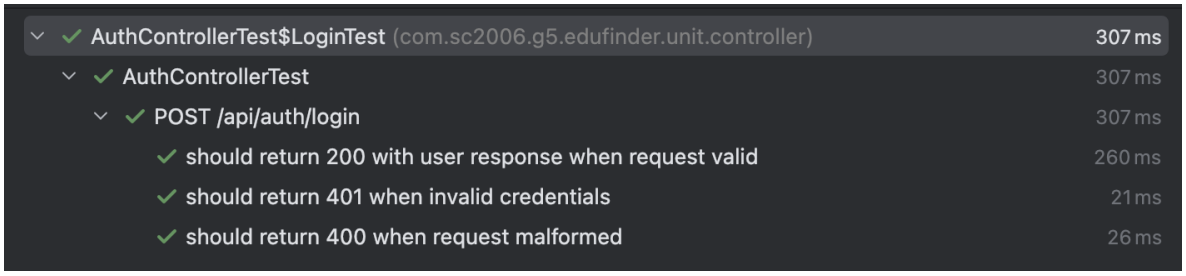


B.1.2 Login

Component: AuthController
Function:
Validates request input (username and password) before delegating to AuthService, and sets the authentication token issued by the service in the response cookie.

Test ID	Input	Equivalence Class	Expected Output
T1	username: valid_username password: AbCd123@	Valid	Status: 200 Token: Issued and Set Body: User Information
T2	{}	Malformed input	Status: 400
T3	password: AbCd123@	Missing Username	Status: 400
T4	username: valid_username	Missing Password	Status: 400
T5	username: valid_username password: wrong_password	Service throw InvalidCredentialsException	Status: 401

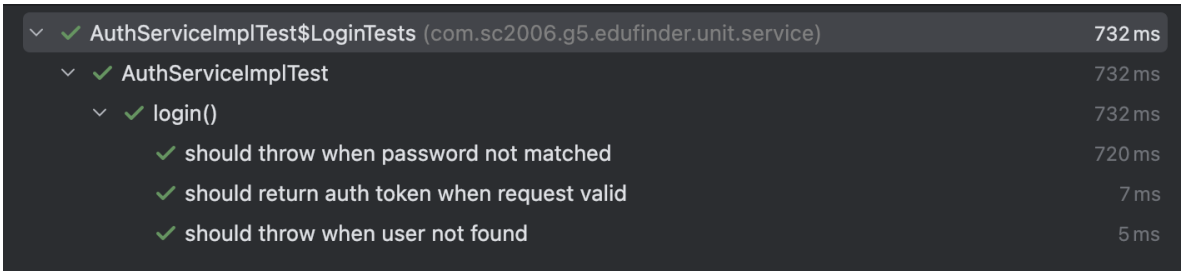
Result:



Component: AuthService
Function:
Check if the username exists, password matches and return a token.

Path ID	Predicate Coverage	Input Condition	Expected Output
P1	UsernameExisted = true PasswordMatched = true	Credential is correct	token with user ID is returned
P2	UsernameExisted = false	Username doesn't exist	InvalidCredentialsException is thrown
P3	UsernameExisted = true PasswordMatched = false	Password doesn't match	InvalidCredentialsException is thrown

Result:



Control Flow Diagram (For Login)