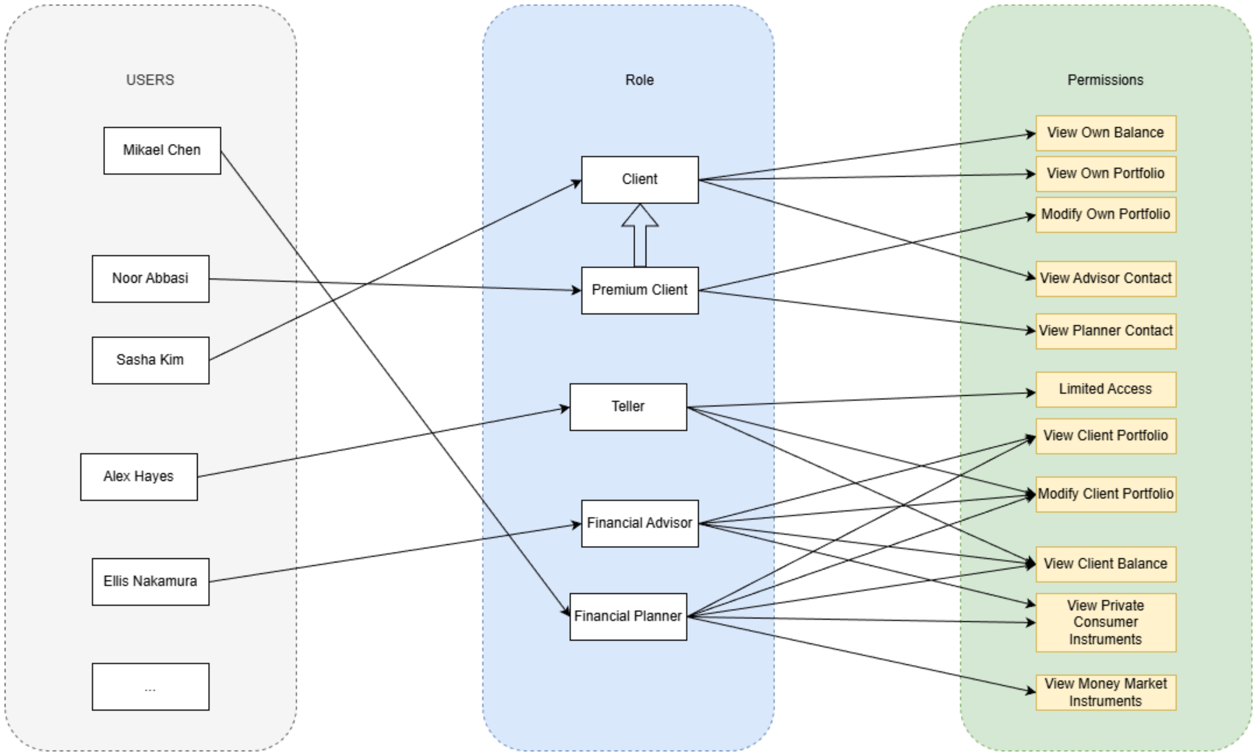
SYSC4810 Assignment

Zachary Gallant 101272210

Problem 1

Chosen acess control model:

a) RBAC: I choose role-based access conrol as the model for justInves't authentication system. Justinvest has multiple user categories with each category having predefined responsibilities. Using RBAC also alows new users to join or change their role without having to change the code or adding responsibility and permissions to each user individualy. When theres a change in polic, only the role definitions needs to be change instead of changing each individual user.



b)

c) The test cases I created are located in file Problem1cTest.java. It contains 20 test, this is what these tests are covering:

- testClientOperations: Tests that Clients can access operations 1, 2, and 4 but can't access operation 3
- testPremiumClientOperations: Makes sure Premium Clients can also do operations 3 and 5, but not operation 6
- testFinancialAdvisorOperations: Tests that Financial Advisors can modify portfolios and view instruments, but can't access operation 6
- testFinancialPlannerOperations: Tests that they have the most permissions including operations 3, 6, and 7
- testTellerOperations: Makes sure Tellers are limited to operations 1 and 2
- testTellerTimeRestrictions: Tests the time-based access by checking that Tellers are denied at 8:59 AM and 5:01 PM, but allowed at 9:00 AM and 4:59 PM
- testInvalidRole: Makes sure that if you try to use an invalid role, it returns false

Problem 2

a) The hash function selection is the SHA-512 with salt. This choice is because it balances security, performance and library suppot. This hash function provides an hash output of 512 bit. Each password uses a unique 16 bytes random salt generated using the java SecureRandom class. This provides 2^128 prossible salt values. This prevents rainbow table attacks. THe salt is stored alongside the has in Base64 encoding inside the passwords.txt file.

b) Zachary_Gallant|3wSxA9OZdZTd5HHzENR6RA==|hdFv/kiqZuB1Z0HlgO9I/NxeUXMmOLoGPue1IXOGN7I//QiIAAOLyCIcpwk4WZi2Ye2i1WiHGdzLt1fgDXko1g==|Financial Advisor

- Username|salt|hashed password|Role
- Username: Username used to login
- Salt: random 128 bit value unique for every user added to the password to prevent rainbow table attacks
- Hashed Password: SHA-512(salt + password), salt is added to the password before hashing and then stored
- Role: RBAC role for the user

d) The test cases I created are located in file Problem2cTest.java. It contains 10 test, this is what these tests are covering:

- testAddUserCreatesEntry: Tests that when I add a user, they actually get stored and I can retrieve them
- testVerifyCorrectPassword: Makes sure that if I provide the right password, it verifies correctly
- testRejectWrongPassword: Tests that the wrong password fails
- testNonexistentUserReturnsNull: Makes sure that if I try to get a user that doesn't exist, it returns null
- testMultipleUsersCanBeStored: Tests that I can add multiple users and they all work independently
- testDifferentSaltsGenerated: This is important for security. Even if two users have the same password, they get different salts so their hashes are different
- testFileFormatIsCorrect: Checks that the file has the right format with pipe delimiters
- testSpecialCharactersInPassword: Tests that passwords with special characters work fine
- testPasswordVerificationIsConsistent: Tests that if I verify the same password twice, I get the same result both times
- testRetrieveUserInfoReturnsThreeComponents: Makes sure the retrieval returns the salt, hash, and role

Problem 3

c) The test cases I created are located in file Problem3Test.java, these are the Password Policy Validation Tests:

- testValidPasswordWithAllRequirements: Tests a password that meets everything
- testPasswordTooShort and testPasswordTooLong: Tests the length requirements (8-12 characters)

- testPasswordMissingUppercase through testPasswordMissingSpecialChar: Tests that all character types are required
- testPasswordMatchesUsername: Makes sure you can't use your username as your password
- testWeakPasswordRejected and testCommonWeakPassword: Tests the blacklist of common passwords
- testSpecialCharsExclamation through testSpecialCharsAmpersand: Tests each of the special characters allowed
- testPasswordWithMultipleSpecialChars: Tests that multiple special chars work
- testPasswordWithMaxLength and testPasswordWithMinLength: Tests the length boundaries

The test cases I created are located in file Problem3Test.java, these are the password Role Validation Tests:

- testRoleClientIsValid through testRoleTellerIsValid: Tests that all 5 valid roles are accepted
- testInvalidRoleRejected: Makes sure fake roles don't work
- testEmptyRoleRejected: Makes sure empty roles are rejected

Problem 4

d) The test cases I created are located in file Problem3Test.java:
   - testLoginWithValidCredentials: Tests that valid username and password work
   - testLoginWithInvalidPassword: Tests that wrong password fails
   - testLoginWithNonexistentUser: Tests that non-existent users can't log in
   - testLoginClientRole through testLoginFinancialPlannerRole: Tests that each role can log in
   - testLoginMultipleUsers: Tests that multiple users can log in with their own credentials
   - testLoginEmptyUsername and testLoginEmptyPassword: Tests that empty fields don't work
   - testLoginCaseSensitiveUsername and testLoginCaseSensitivePassword: Tests that passwords and usernames are case-sensitive
   - testLoginWithSpecialCharPassword: Tests that special characters in passwords work
   - testLoginFailsWithPartialMatch: Makes sure partial usernames don't work
   - testLoginConsistency: Tests that logging in multiple times with the same credentials work each time