Testing Documentation

Test Containers were used for the DB interaction and Greenmail as a test email server for the email functionality. Further testing would require a test environment.

Unit Tests

Services

Test Case ID	Description	Test Steps	Test Method	Expected Output	Actual Output
TC_01	Successful user registration	1) Create a valid register request 2) Create a user based on the register request 3) Mock the encoder to return a hashed password 4) Mock the JWT service to return a token 5) Mock the user service 6) Call the register method	AuthServiceTest shouldRegisterUser()	1) JWT token 2) User service register() method called once	As expected
TC_02	Successful user login	1) Create a valid login request 2) Mock the authentication manager to successfully authenticate 3) Mock the JWT service to return a token 4) Call the login method	AuthServiceTest shouldAuthenticateUser()	1) JWT token	As expected
TC_03	Failed login request when wrong credentials are provided	1) Create a valid login request 2) Mock the authentication manager to throw an exception when attempting to authenticate 3) Call	AuthServiceTest shouldThrow UnauthorizedException WhenAuthEmailOr PasswordIsWrong()	1) Unauthorized Exception with the relevant error message	As expected

		the login method			
TC_04	Successful JWT token assignment	1) Create a JWT token 2) Mock the JWT encoder to return the predefined token 3) Call the assign token method	JwtServiceTest shouldAssignToken()	1) JWT token	As expected
TC_05	Successful password reset token creation	1) Register a user 2) Create password reset request with existing email 3) Mock the email service 4) Call create password reset token method	PasswordResetServiceTest shouldCreatePassword ResetTokenWhenUser IsFound()	1) Email sendPasswordResetE mail() method called once	As expected
TC_06	Failed password reset token creation when provided email doesn't exist in our DB	1) Register a user 2) Create password reset request with different email 3) Call create password reset token method	PasswordResetServiceTest shouldNotCreate PasswordResetToken WhenUserIsNotFound()	1) Email service was not called	As expected
TC_07	Failed password reset token validation for blank or malformed token	1) Create password reset request 2) Call reset password method	PasswordResetServiceTest shouldNotResetPassword WhenPasswordReset TokenIsBlankOr Malformed()	1) False	As expected
TC_08	Failed password reset token validation for expired token	1) Register a user 2) Create a password reset token and store it in the db 3) Create password reset request 4) Call reset password method	PasswordResetServiceTest shouldNotResetPassword WhenPasswordReset TokenExpired ()	1) False	As expected
TC_09	Invalidate all previous tokens when new password reset token is requested	1) Create a password reset token 2) Create a new Password reset request 3) Call create password reset token method	PasswordResetServiceTest shouldInvalidateAllPreviousT okensWhenNewResetTokenI sGenerated()	1) The initial token is not present in the database	As expected
TC_10	Successful	1) Register a user 2)	PasswordResetServiceTest	1) True 2)The email	As

	password reset	Create a password reset token and save it to the db 3) Mock the email service 4) Create a Password reset confirmation request with the initial pre hashed token 5) Call reset password method	shouldResetPassword()	service sendPasswordReseS uccessEmail() method called once 3) The provided token is not present in the database meaning the password was successfully updated and the token was deleted	expected
TC_11	Successful user registration	1) Generate the user 2) Call the register method	UserServiceTest shouldRegisterUser()	1) User is found after registration	As expected
TC_12	Failed to register user with email that already exists	1) Register a user 2) Create a user with the same email 3) Call the register method	UserServiceTest shouldThrow DuplicateResource ExceptionIfEmail AlreadyExists()	1) Duplicate Resource Exception with relevant error message	As expected
TC_13	Failed to register user with username that already exists	1) Register a user 2) Create a user with the same username 3) Call the register method	UserServiceTest shouldThrow DuplicateResource ExceptionIfUsername AlreadyExists()	1) Duplicate Resource Exception with relevant error message	As expected
TC_14	Failed user validation when first name exceeds the max length defined in the DB column	1) Create a user with first name length greater than 30 characters 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenFirstname Exceeds MaxLength()	1) Illegal Argument Exception with relevant error message	As expected
TC_15	Failed user validation when first name contains numbers	1) Create a user with first name containing numbers 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenFirstname ContainsNumbers()	1) Illegal Argument Exception with relevant error message	As expected
TC_16	Failed user validation when first name contains special characters	1) Create a user with first name containing special characters 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenFirstname ContainsSpecial Characters()	1) Illegal Argument Exception with relevant error message	As expected
TC_17	Failed user validation when	1) Create a user with last name	UserServiceTest shouldThrow	1) Illegal Argument Exception with	As expected

	last name exceeds the max length defined in the DB column	length greater than 30 characters 2) Call the validation method	IllegalArgument ExceptionWhen Lastname Exceeds MaxLength()	relevant error message	
TC_18	Failed user validation when last name contains numbers	1) Create a user with last name containing special numbers 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenLastname ContainsNumbers()	1) Illegal Argument Exception with relevant error message	As expected
TC_19	Failed user validation when last name contains special characters	1) Create a user with last name containing special characters 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenLastname ContainsSpecial Characters()	1) Illegal Argument Exception with relevant error message	As expected
TC_20	Failed user validation when username exceeds the max length defined in the DB column	1) Create a user with username length greater than 30 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenUsername Exceeds MaxLength()	1) Illegal Argument Exception with relevant error message	As expected
TC_21	Failed user validation when email exceeds the max length defined in the DB column	1) Create a user with email length greater than 50 2) Call the validation method.	UserServiceTest shouldThrow IllegalArgument ExceptionWhenEmail ExceedsMaxLength()	1) Illegal Argument Exception with relevant error message	As expected
TC_22	Failed user validation when email doesn't contain '@' symbol	1) Create a user with email with no '@'s symbol 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenEmail DoesNotContainAtSymbol()	1) Illegal Argument Exception with relevant error message	As expected
TC_23	Failed user validation when bio exceeds the max length defined in the DB column	1) Create a user with bio length greater than 150 characters 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenBio Exceeds MaxLength()	1) Illegal Argument Exception with relevant error message	As expected
TC_24	Failed user validation when location exceeds the max length defined in the	1) Create a user with location length greater than 50 characters 2) Call the validation method	UserServiceTest shouldThrow IllegalArgument ExceptionWhenLocation Exceeds MaxLength()	1) Illegal Argument Exception with relevant error message	As expected

	DB column				
TC_25	Failed user validation when company exceeds the max length defined in the DB column	1) Create a user with company length greater than 50 characters 2) Call the validation method.	UserServiceTest shouldThrow IllegalArgument ExceptionWhenCompany Exceeds MaxLength()	1) Illegal Argument Exception with relevant error message	As expected
TC_26	Successful user retrieval	1) Register a user 2) Create a userDTO based on the registered user 3) Mock the JWT service to return the userId of our registered user 4) Call the get user method	UserServiceTest shouldGetUser()	1) UserDTO that matches the details of the retrieved user	As expected
TC_27	Failed to retrieve user when user is not found	1) Mock the JWT service to return a userId of a user that doesn't exist 2) Call the get user method	UserServiceTest shouldThrowResourceNot FoundException WhenUserIsNotFound()	1) Resource Not Found Exception with relevant error message	As expected
TC_28	Successfully getting user profile	1) Register a user 2) Mock the JWT service to return the userId of our registered user 3) Create a user profile 4) Call the get profile method	UserServiceTest shouldGet UserProfile()	1) The retrieved profile matches the created one	As expected
TC_29	Failed getting user profile when user is not found	1) Mock the JWT service to return a userId of a user that doesn't exist 2) Call the get profile method	UserServiceTest shouldThrowResource NotFoundExceptionWhenUs erIsNotFoundToGet Profile()	1) Resource Not Found Exception with relevant error message	As expected
TC_30	Successful user profile update when user is found and the update profile request is valid	1) Register a user 2) Create a valid user profile update request 3) Call the update profile method	UserServiceTest shouldUpdate UserProfile()	1) User profile update request values match the updated user profile in the database	As expected
TC_31	Failed to update user profile with	1) Register a user 2) Create a valid user profile update	UserServiceTest shouldThrowDuplicate ResourceExceptionWhe	1) Duplicate Resource Exception with relevant error	As expected

	username that already exists	request 3) Mock the JWT service to return the userId of our registered user 4) Call the update user profile method	UsernameAlreadyExists ForProfileUpdate Request()	message	
TC_32	Failed user profile update request when user is not found	1) Create a valid user profile update request 2) Mock the JWT service to return a userId of a user that doesn't exist 3) Call the update profile method	UserServiceTest shouldThrowResource NotFoundExceptionWhenUs erIsNotFoundToUpdateProfil e()	1) Resource Not Found Exception with relevant error message	As expected
TC_33	Successful user password update when user is found and the update password request is valid	1) Register a user 2) Create a valid user profile update request 3) Mock the JWT service to return the userId of our registered user 4) Call the update password method	UserServiceTest shouldUpdateUser Password()	1) The updated hashed password matches the password in our DB	As expected
TC_34	Failed user password update request when old password is wrong	1) Register a user 2) Create a valid user password update request with wrong old password	UserServiceTest shouldThrowBadCredentials ExceptionWhenOld PasswordIsWrong()	1) Bad Credentials Exception with relevant error message	As expected
TC_35	Failed user password update request when user is not found	1) Create a valid user password update request 2) Mock the JWT service to return a userId of a user that doesn't exist 3) Call the update password method	UserServiceTest shouldThrowResource NotFoundExceptionWhenUs erIsNotFoundToUpdatePass word()	1) Resource Not Found Exception with relevant error message	As expected
TC_36	Successful email update token creation	1) Register a user 2) Create a valid email update request 3) Mock the JWT service to return the userId of our registered user 4) Mock the email	UserServiceTest shouldCreateEmail UpdateToken()	1) Email sendEmailVerificatio nl() method called once	As expected

		service 5) Call the create email update token method			
TC_37	Failed user email update request when password is wrong	1) Register a user 2) Create an email update request with wrong password 3) Mock the JWT service to return the userId of our registered user 4) Call the create email update token method	UserServiceTest shouldThrowBad CredentialsExceptionWhen PasswordIsWrongFor EmailUpdate Request()	1) Bad Credentials Exception with relevant error message	As expected
TC_38	Failed user email updated request when with new email that already exists	1) Register a user 2) Create an email update request with an existing email 3) Mock the JWT service to return the userId of our registered user 4) Call the create email update token method	UserServiceTest shouldThrowDuplicate ResourceExceptionWhen NewEmailAlreadyExists ForEmailUpdateRequest()	1) Duplicate Resource Exception with relevant error message	As expected
TC_39	Failed email update token validation for blank or malformed token	1) Create a user email update request 2) Call the update email method	UserServiceTest shouldNotUpdateEmail WhenEmailUpdateToken IsBlankOrMalformed()	1) False	As expected
TC_40	Failed email update token validation for expired token	1) Create a user email update request 2) Call update email method	UserServiceTest shouldNotUpdateEmail WhenEmailUpdateToken Expired ()	1) False	As expected
TC_41	Failed user email update request when user is not found	1) Create a user email update request 2) Mock the JWT service to return a userId of a user that doesn't exist 3) Call the create email update token method	UserServiceTest shouldThrowResource NotFoundExceptionWhen UserIsNotFoundTo UpdateEmail()	1) Resource Not Found Exception with relevant error message	As expected

TC_42	Invalidate all previous tokens when new email update token is requested	1) Register a user 2) Create an email update token 3) Create a new email update request 4) Mock the JWT service to return the userId of our registered user 5) Call create email update token method	UserServiceTest shouldInvalidateAll PreviousTokensWhenNewE mailUpdateToken IsGenerated()	1) The initial token is not present in the database	As expected
TC_43	Successful email update	1) Register a user 2) Create an email update token 3) Call update email method	UserServiceTest shouldUpdateEmail()	1) The provided token is not present in the database meaning the email was successfully updated and the token was deleted	As expected
TC_44	Failed getting user history when from date is null or empty	1) Register a user 2) Create only a to date 3) Mock the JWT service to return the userId of our registered user 4) Call the get history method	UserServiceTest shouldThrowIllegal ArgumentExceptionWhen FromIsNullOrEmpty()	1) Illegal Argument Exception with relevant error message	As expected
TC_45	Failed getting user history when to date is null or empty	1) Register a user 2) Create only a from date 3) Mock the JWT service to return the userId of our registered user 4) Call the get history method	UserServiceTest shouldThrowIllegal ArgumentExceptionWhenTo IsNullOrEmpty()	1) Illegal Argument Exception with relevant error message	As expected
TC_46	Failed getting user history when at least one date is not in the supported format (yyyy-MM-dd)	1) Register a user 2) Create one valid and one invalid date 3) Mock the JWT service to return the userId of our registered user 4) Call the get history method	UserServiceTest shouldThrowIllegal ArgumentExceptionWhenAt LeastOneDateIsNotInThe SupportedFormat	1) Illegal Argument Exception with relevant error message	As expected

TC_47	Successful user account deletion	1) Register a user 2) Create user account delete request 3) Mock the JWT service to return the userId of our registered user 4) Call the delete account method	UserServiceTest shouldDeleteAccount()	1) User profile is not found after account deletion	As expected
TC_48	Failed user account deletion when the provided password is wrong	1) Register a user 2) Create user account delete request with wrong password3) Mock the JWT service to return the userId of our registered user 4) Call the delete account method	UserServiceTest shouldThrowBad CredentialsException WhenPasswordIsWrong ForAccountDeletion()	1) Bad Credentials Exception with relevant error message	As expected
TC_49	No filtering when no constraints provided	1) Create a list of analysis reports 2) Create an empty constraints list 3) Call assess analysis result	AssessmentServiceTest shouldNotFilterReports WhenNoConstraints AreProvided()	1) No interactions with the filtering service	As expected
TC_50	Empty compliant list when all reports are non-compliant with the constraints	1) Create a list of analysis reports 2) Create a list of constraints 3) Call the filter method	FilteringServiceTest shouldReturnAnEmpty CompliantListWhenAll ReportsAreNonCompliant WithTheConstraints()	1) Compliant list is empty 2) Non-compliant list has the size of the total reports	As expected
TC_51	We have both compliant and non-compliant reports	1) Create a list of analysis reports 2) Create a list of constraints 3) Call the filter method	FilteringServiceTest shouldReturnReports InBothLists()	1) Compliant list contains reports 2) Non-compliant list contains reports	As expected
TC_52	Empty non-compliant list when all reports are compliant with the constraints	1) Create a list of analysis reports 2) Create a list of constraints 3) Call the filter method	FilteringServiceTest shouldReturnAnEmpty NonCompliantListWhenAll ReportsAreCompliant WithTheConstraints()	1) Non-compliant list is empty 2) Compliant list has the size of the total reports	As expected
TC_53	Ranking reports when no	1) Create an empty preferences list 2)	RankingServiceTest shouldRankReportsWhen	1) The actual rank is the equal to	As expected

	preferences are provided	Create a quality metric report map 3) Call the rank tree method	NoPreferencesAre Provided()	expected one	
TC_54	Ranking reports when preferences are provided	1) Create a preferences list 2) Create a quality metric report map 3) Call the rank tree method	RankingServiceTest shouldRankReportsWhen PreferencesAre Provided()	1) The actual rank is the equal to expected one	As expected
TC_55	Successfully building the rank tree	1) Call the build tree method	TreeServiceTest shouldBuildTree()	1) All the parent nodes have the correct child nodes	As expected
TC_56	Failed analysis request when the sum of the weights of the child nodes is greater than 1	1) Call the build tree method 2) Create a list of preferences where the sum of the weights of the child nodes is greater than 1	TreeServiceTest shouldThrowIllegal ArgumentExceptionWhen TheSumOfTheWeights OfTheChildNodesIsGreater ThanOne()	1) Illegal Argument Exception with relevant error message	As expected
TC_57	Failed analysis request when the sum of the weights of all the child nodes is less than 1	1) Call the build tree method 2) Create a list of preferences where the sum of the weights of the child nodes is less than 1	TreeServiceTest shouldThrowIllegal ArgumentExceptionWhen TheSumOfTheWeights ForAllChildNodesIsLess ThanOne()	1) Illegal Argument Exception with relevant error message	As expected
TC_58	Successful analysis creation	1) Register a user 2) Create a list of preferences 3) Create a list of constraints 4) Create a list of analysis reports 5) Call save analysis process	AnalysisServiceTest shouldSaveAnalysis Process()	1) The auto generated id from the database is a positive number	As expected
TC_59	Successfully getting user history	1) Register a user 2) Create a list of preferences 3) Create a list of constraints 4) Create a list of analysis reports 5) Call save analysis process 6) Call the get history method	AnalysisServiceTest shouldGetUser History()	1) The returning list of analyses is equal to 1	As expected

TC_60	Successful analysis deletion	1) Register a user 2) Create a list of preferences 3) Create a list of constraints 4) Create a list of analysis reports 5) Call save analysis process 6) Call the delete analysis method	AnalysisServiceTest shouldDelete Analysis()	1) Resource Not Found Exception with relevant error message	As expected
TC_61	Successfully getting analysis request	1) Register a user 2) Create a list of preferences 3) Create a list of constraints 4) Create a list of analysis reports 5) Call save analysis process 6) Call the get analysis request method	AnalysisServiceTest shouldGetAnalysis Request()	1) Retrieved project urls, preferences and constraints are the same with the initial ones	As expected
TC_62	Successfully detecting languages	1) Mock the docker service to return a language with a percentage 2) Call the detect languages method	LanguageServiceTest shouldDetectLanguages()	1) The returned map contains 1 entry with the language as key and the percentage as value	As expected
TC_63	Successfully identifying supported languages	1) Create a map with entries that at least 1 is a supported language 2) Call verify supported languages method`	LanguageServiceTest shouldReturnTrue WhenAtLeastOne DetectedLanguage IsSupported()	1) True	As expected
TC_64	No supported languages detected	1) Create a map with entries that have no supported languages 2) Call verify supported languages method	LanguageServiceTest shouldReturnFalse WhenNoneOfThe DetectedLanguage IsSupported()	1) False	As expected
TC_65	Successfully applying utility functions for each quality metric	1) Create a quality metric report map with key the quality metric and value the expected value after applying the	MetricServiceTest shouldApplyUtf()	1) The returned map has the same key value pairs as the initial one	As expected

		utf 2) Call apply utf method			

Mappers

Test Case ID	Description	Test Steps	Test Method	Expected Output	Actual Output
TC_01	Successfully mapping a user to a userDTO	1) Create a user 2) Call the apply method	UserDTOMapperTest shouldMapUser ToUserDTO()	1) Initial user and UserDTO have the same values on the same properties	As expected
TC_02	Successfully mapping an analysis report record to an analysis report object	1) Create an analysis report object 2) Mock the result set to return an analysis report record 3) Call the map row method	AnalysisReportRow MapperTest shouldMapRowTo AnalysisReport()	1) Initial analysis report and the row mapped one have the same values as	As expected
TC_03	Successfully mapping an analysis record to an analysis object	1) Create an analysis object 2) Mock the ResultSet to return an analysis record 3) Call the map row method	AnalysisRowMapperTest shouldMapRowTo Analysis()	1) Initial analysis and the row mapped one have the same values	As expected
TC_04	Successfully mapping a constraint record to a constraint object	1) Create a constraint object 2) Mock the result set to return a constraint record	ConstraintRowMapperTest shouldMapRowTo Constraint()	1) Initial constraint and the row mapped one have the same values	As expected
TC_05	Successfully mapping an email update token record to an email update token object	1) Mock the result set to return an email update token record	EmailUpdateTokenRow MapperTest shouldMapRowTo EmailUpdate Token()	1) Email update token object returned from row mapper has the expected values	As expected
TC_06	Successfully mapping a password reset token record to a password reset token object	1) Mock the result set to return a password reset token record	PasswordResetTokenRow MapperTest shouldMapRowTo PasswordReset Token()	1) Password reset token object returned from row mapper has the expected values	As expected

TC_07	Successfully mapping a preference record to a preference object	1) Create a preference object 2) Mock the result set to return a preference record	PreferenceRowMapper Test shouldMapRow ToPreference()	1) Initial preference and the row mapped one have the same values	As expected
TC_08	Successfully mapping a user record to a user object	1) Create a user object 2) Mock the result set to return a user record	UserRowMapperTest shouldMapRow ToUser()	1) Initial user and the row mapped one have the same values	As expected

Deserializers

Test Case ID	Description	Test Steps	Test Method	Expected Output	Actual Output
TC_01	Successfully deserializing a string representation of a quality attribute to its corresponding enum value	1) Create a string representation of a quality attribute 2) Mock the parser to return the initial string 3) Call the deserialize method	QualityAttribute DeserializerTest shouldDeserializeQuality Attribute()	1) The deserialized quality attribute enum matches the initial string representation	As expected
TC_02	Successfully deserializing a string representation of a quality attribute ignoring case and extra spaces to its corresponding enum value	1) Create a string representation of a quality attribute with mixed cases and extra spaces 2) Mock the parser to return the initial string 3) Call the deserialize method	QualityAttribute DeserializerTest shouldDeserializeQuality AttributeIgnoringCase AndExtraSpaces()	1) The deserialized quality attribute enum matches the initial string representation ignoring case and extra spaces	As expected
TC_03	Failed quality attribute deserialization for empty, invalid string input	1) Mock the parser to return the initial string 2) Call the deserialize method	QualityAttribute DeserializerTest shoudlThrowIllegal ArgumentExceptionWhen QualityAttribute IsInvalid()	1) Illegal Argument Exception with relevant error message	As expected
TC_04	Successfully deserializing a string representation	1) Create a string representation of a quality metric 2) Mock the parser to	QualityMetricDeserializer Test shouldDeserializeQuality Metric()	1) The deserialized quality metric enum matches the initial string	As expected

	of a quality metric to its corresponding enum value	return the initial string 3) Call the deserialize method		representation	
TC_05	Successfully deserializing a string representation of a quality metric ignoring case and extra spaces to its corresponding enum value	1) Create a string representation of a quality metric with mixed cases and extra spaces 2) Mock the parser to return the initial string 3) Call the deserialize method	QualityMetric DeserializerTest shouldDeserializeQuality MetricIgnoringCase AndExtraSpaces()	1) The deserialized quality metric enum matches the initial string representation ignoring case and extra spaces	As expected
TC_06	Failed quality metric deserialization for empty, invalid string input	1) Mock the parser to return the initial string 2) Call the deserialize method	QualityMetric DeserializerTest shoudlThrowIllegal ArgumentExceptionWhen QualityMetric IsInvalid()	1) Illegal Argument Exception with relevant error message	As expected
TC_07	Successfully deserializing a string representation of an operator symbol to its corresponding Quality metric operator enum value	1) Create a string representation of an operator symbol 2) Mock the parser to return the initial string 3) Call the deserialize method	QualityMetricOperator DeserializerTest shouldDeserializeSymbolTo QualityMetricOperator()	1) The deserialized operator's symbol matches the initial symbol	As expected
TC_08	Failed quality metric operator deserialization for empty, invalid string input	1) Mock the parser to return the initial string 2) Call the deserialize method	QualityMetricOperator DeserializerTest shoudlThrowIllegal ArgumentExceptionWhen SymbolIsInvalid()	1) Illegal Argument Exception with relevant error message	As expected

Controllers

Test Case ID	Description	Test Steps	Test Method	Expected Output	Actual Output
TC_01	401 Unauthorized: When	1) Construct a JSON payload for the analysis request 2)	AnalysisControllerTest shouldReturnHTTP401 WhenAnalyzeIsCalled	1) Status UNAUTHORIZED 2) No interactions with	As expected

	unauthenticated user calls the analyze endpoint	Perform a POST request to the analyze endpoint	ByUnauthenticated User()	the analysis service	
TC_02	400 Bad Request: When authenticated user provides an empty list of repositories	1) Construct a JSON payload for the analysis request with an empty projects urls list 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenListOfProjectUrls IsEmpty()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_03	400 Bad Request: When authenticated user provides null for project urls	1) Construct a JSON payload for the analysis request where project urls list is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenListOfProjectUrls IsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_04	400 Bad Request: When authenticated user provides null for a constraint quality metric	1) Construct a JSON payload for the analysis request where a constraint quality metric is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenConstraintQuality MetricIsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_05	400 Bad Request: When authenticated user provides null for a constraint quality metric operator	1) Construct a JSON payload for the analysis request where a constraint quality metric operator is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenConstraintQuality MetricOperator IsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_06	400 Bad Request: When authenticated user provides null for a constraint threshold	1) Construct a JSON payload for the analysis request where a constraint threshold is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenConstraintThreshold IsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_07	400 Bad	1) Construct a JSON	AnalysisControllerTest	1) Status	As

	Request: When authenticated user provides invalid values for a constraint threshold	payload for the analysis request where a constraint threshold has invalid values 2) Perform a POST request to the analyze endpoint	shouldReturnHTTP400 WhenConstraintThreshold IsInvalid()	BAD_REQUEST 2) No interactions with the analysis service	expected
TC_08	400 Bad Request: When authenticated user provides null for a preference quality attribute	1) Construct a JSON payload for the analysis request where a preference quality attribute is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenPreferenceQuality AttributeIsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_09	400 Bad Request: When authenticated user provides null for a preference weight	1) Construct a JSON payload for the analysis request where a preference weight is null 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenPreferenceWeight IsNull()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_10	400 Bad Request: When authenticated user provides invalid values for a preference weight	1) Construct a JSON payload for the analysis request where a preference weight has invalid values 2) Perform a POST request to the analyze endpoint	AnalysisControllerTest shouldReturnHTTP400 WhenPreferenceWeight IsInvalid()	1) Status BAD_REQUEST 2) No interactions with the analysis service	As expected
TC_11	401 Unauthorized: When unauthenticated user calls analysis result endpoint	1) Perform a GET request to analysis result endpoint	AnalysisControllerTest shouldReturnHTTP401When GetAnalysisResultIsCalled ByUnauthenticated User()	1) Status UNAUTHORIZED 2) No interactions with the analysis service	As expected
TC_12	401 Unauthorized: When unauthenticated user calls the refresh request	1) Construct a JSON payload for the refresh request 2) Perform a PUT request to the refresh request	AnalysisControllerTest shouldReturnHTTP401When RefreshAnalysisResult IsCalled ByUnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the analysis service	As expected

	endpoint	endpoint			
TC_13	401 Unauthorized: When unauthenticated user calls delete analysis endpoint	1) Perform a DELETE request to the analysis endpoint	AnalysisControllerTest shouldReturnHTTP401WhenDeleteAnalysisIsCalledByUnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the analysis service	As expected
TC_14	401 Unauthorized: When unauthenticated user calls analysis request endpoint	1) Perform a GET request to the analysis request endpoint	AnalysisControllerTest shouldReturnHTTP401WhenGetAnalysisRequesttIsCalledByUnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the analysis service	As expected
TC_15	201 Created, Jwt token in the respone body: Successful user registration	1) Construct a JSON payload for the register request 2) Construct a JSON payload with the expected response body 3) Mock the authentication service to return a jwt token 4) Perform a POST request to the signup endpoint	AuthControllerTest shouldReturnJwtTokenAndHTTP201WhenUserIsRegisteredSuccessfully()	1) Status CREATED 2) The response body contains the token returned from the auth service	As expected
TC_15	400 Bad Request: Failed user registration when first name is null or empty	1) Construct a JSON payload for the register request where firstname is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the signup endpoint	AuthControllerTest shouldReturnHTTP400WhenRegisterFirstnameIsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_16	400 Bad Request: Failed user	1) Construct a JSON payload for the register request	AuthControllerTest shouldReturnHTTP400WhenRegisterLastname	1) Status BAD_REQUEST 2) The response body	As expected

	registration when last name is null or empty	where lastname is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the signup endpoint	IsNullOrEmpty()	contains relevant error message	
TC_17	400 Bad Request: Failed user registration when username is null or empty	1) Construct a JSON payload for the register request where username is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the signup endpoint	AuthControllerTest shouldReturnHTTP400 WhenRegisterUsername IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_18	400 Bad Request: Failed user registration when email is null or empty	1) Construct a JSON payload for the register request where email is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the signup endpoint	AuthControllerTest shouldReturnHTTP400 WhenRegisterEmail IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_19	400 Bad Request: Failed user registration when password is null or empty	1) Construct a JSON payload for the register request where password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the signup endpoint	AuthControllerTest shouldReturnHTTP400 WhenRegisterPassword IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected

TC_20	200 OK, Jwt token in the respone body: Successful user login	1) Construct a JSON payload for the login request 2) Construct a JSON payload with the expected response body 3) Mock the authentication service to return a jwt token 4) Perform a POST request to the login endpoint	AuthControllerTest shouldReturnJwtToken AndHTTP200WhenUserIs LoggedInSuccessfully()	1) Status OK 2) The response body contains the token returned from the auth service	As expected
TC_21	400 Bad Request: Failed user login when email is null or empty	1) Construct a JSON payload for the login request where email is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the login endpoint	AuthControllerTest shouldReturnHTTP400 WhenLoginEmail IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_22	400 Bad Request: Failed user login when password is null or empty	1) Construct a JSON payload for the login request where password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the login endpoint	AuthControllerTest shouldReturnHTTP400 WhenLoginPassword IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_23	202 Accepted: When user makes a password reset request	1) Construct a JSON payload for the password reset request 2) Mock the password reset service 3) Perform a POST request to the password reset endpoint	AuthControllerTest shouldReturnHTTP202 ForPasswordResetRequest RegardlessIfTheEmail Exists()	1) Status ACCEPTED	As expected

TC_24	400 Bad Request: Failed password reset request when email is null or empty	1) Construct a JSON payload for the password reset request where email is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the password reset endpoint	AuthControllerTest shouldReturnHTTP400 WhenPasswordResetEmail IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_25	401 Unauthorized: Failed password reset when token invalid (empty, malformed, expiry)	1) Construct a JSON payload for the password reset confirmation request where token is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the password reset confirm endpoint	AuthControllerTest shouldReturnHTTP401 WhenPasswordReset ConfirmationToken IsInvalid()	1) Status UNAUTHORIZED 2) The response body contains relevant error message	As expected
TC_26	400 Bad Request: Failed password reset when new password is null or empty	1) Construct a JSON payload for the password reset confirmation request where password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the password reset confirm endpoint	AuthControllerTest shouldReturnHTTP400 WhenPasswordReset ConfirmationPassword IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_27	400 Bad Request: Failed password reset when new password does not meet the	1) Construct a JSON payload for the password reset confirmation request where password does not	AuthControllerTest shouldReturnHTTP400 WhenPasswordReset ConfirmationPassword DoesNotMeetThe Requirements()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected

	requirements	meet the requirements 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the password reset confirm endpoint			
TC_28	204 No Content: Successful password reset	1) Construct a JSON payload for the password reset confirmation request 2) Perform a PUT request to the password reset confirm endpoint	AuthControllerTest shouldReturnHTTP204WheTokenAndNewPasswordAreValid()	1) Status NO_CONTENT	As expected
TC_29	200 OK, UserDTO in the response body: Successfully getting user	1) Create a UserDTO object 2) Construct a JSON payload with the expected response body 3) Mock the user service to return the DTO 4) Perform a GET request to the user endpoint	UserControllerTest shouldReturnUserDTOAndHTTP200()	1) Status OK 2) The response body has the same values as our initial UserDTO	As expected
TC_30	404 Not found: When user is not found	1) Construct a JSON payload with the expected response body 2) Mock the user service to throw a Not Found exception 3) Perform a GET request to the user endpoint	UserControllerTest shouldThrowNotFoundExceptionWhenUserIsNotFound()	1) Status NOT_FOUND 2) The response body contains relevant error message	As expected
TC_31	401 Unauthorized: When unauthenticated user calls get user	1) Perform a GET request to the user endpoint	UserControllerTest shouldReturnHTTP401WhenGetUserIsCalledByUnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_32	200 OK, UserProfile in the response	1) Create a User Profile object 2) Construct a JSON	UserControllerTest shouldReturnUserProfileAndHTTP200()	1) Status OK 2) The response body has the same values as	As expected

	body: Successfully getting user profile	payload with the expected response body 3) Mock the user service to return the profile 4) Perform a GET request to the user profile endpoint		our initial UserProfile object	
TC_33	404 Not found: When user is not found to get profile	1) Construct a JSON payload with the expected response body 2) Mock the user service to throw a Resource Not Found exception 3) Perform a GET request to the update profile endpoint	UserControllerTest shouldReturnHTTP404 WhenUserIsNotFound ToGetProfile()	1) Status NOT_FOUND 2) The response body contains relevant error message	As expected
TC_34	401 Unauthorized: When unauthenticated user calls get profile	1) Perform a GET request to the user profile endpoint	UserControllerTest shouldReturnHTTP401When GetProfileIsCalledBy UnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_35	204 No Content: Successfully updating user profile	1) Construct a JSON payload for the user profile update request 2) Mock the user service 3) Perform a PUT request to the update profile endpoint	UserControllerTest shouldReturnHTTP204When ProfileIsUpdated()	1) Status NO_CONTENT 2) User service update profile method was called once	As expected
TC_36	409 Conflict: Failed user profile update when the provided username already exists	1) Construct a JSON payload for the user profile update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw a Duplicate Resource exception 4) Perform a PUT request to the	UserControllerTest shouldReturnHTTP409 WhenUpdatingProfileWith ExistingUsername()	1) Status CONFLICT 2) The response body contains relevant error message	As expected

		update profile			
TC_37	404 Not found: When user is not found to update profile	1) Construct a JSON payload for the user profile update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw a Resource Not Found exception 4) Perform a PUT request to the update profile endpoint	UserControllerTest shouldReturnHTTP404 WhenUserIsNotFound ToUpdateProfile()	1) Status NOT_FOUND 2) The response body contains relevant error message	As expected
TC_38	401 Unauthorized: When unauthenticated user calls update profile	1) Construct a JSON payload for the user profile update request 2) Perform a PUT request to the update profile endpoint	UserControllerTest shouldReturnHTTP401When UpdateProfileIsCalledBy UnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_39	204 No Content: Successful user password update	1) Construct a JSON payload for the user password update request 2) Mock the user service 3) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP204When PasswordIsUpdated()	1) Status NO_CONTENT 2) User service update password method was called once	As expected
TC_40	400 Bad Request: Failed password update when old password is null or empty	1) Construct a JSON payload for the user password update request where old password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP400 WhenOldPassword IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message 3) No interactions with the user service	As expected

TC_41	400 Bad Request: Failed password update when old password is wrong	1) Construct a JSON payload for the user password update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw a Bad Credentials exception M3) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP400 WhenOldPassword IsWrong ()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_42	400 Bad Request: Failed password update when new password is null or empty	1) Construct a JSON payload for the user password update request where new password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP400 WhenNewPassword IsNullOrEmpty()	1) Status BAD_REQUEST 2) The response body contains relevant error message 3) No interactions with the user service	As expected
TC_43	400 Bad Request: Failed password update when new password does not meet the requirements	1) Construct a JSON payload for the user password update request 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP400 WhenNewPasswordDoes NotMeetRequeirments()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_44	404 Not found: When user is not found to update password	1) Construct a JSON payload for the user password update request 2) Construct a JSON payload with the expected response	UserControllerTest shouldReturnHTTP404 WhenUserIsNotFound ToUpdatePassword()	1) Status NOT_FOUND 2) The response body contains relevant error message	As expected

		body 3) Mock the user service to throw a Resource Not Found exception 4) Perform a PUT request to the update password endpoint			
TC_45	401 Unauthorized: When unauthenticated user calls update password	1) Construct a JSON payload for the user password update request 2) Perform a PUT request to the update password endpoint	UserControllerTest shouldReturnHTTP401WhenUpdatePasswordIsCalledByUnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_46	202 Accepted: Sending a link to the new user's email for email update	1) Construct a JSON payload for the user email update request 2) Mock the user service 3) Perform a POST request to the update email endpoint	UserControllerTest shouldReturn202WhenEmailTokenIsCreatedForEmailUpdate()	1) Status ACCEPTED 2) User service create email update token method was called once	As expected
TC_49	400 Bad Request: Failed email update when password is null or empty	1) Construct a JSON payload for the user email update request where password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the update email endpoint	UserControllerTest shouldReturnHTTP400WhenPasswordIsNullOrEmptyForEmailUpdate()	1) Status BAD_REQUEST 2) The response body contains relevant error message 3) No interactions with the user service	As expected
TC_50	400 Bad Request: Failed to update email when provided password is wrong	1) Construct a JSON payload for the user email update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to	UserControllerTest shouldReturnHTTP400WhenWrongPasswrodIsProvidedForEmailUpdate()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected

		throw a Bad Credentials exception 4) Perform a POST request to the update email endpoint			
TC_51	400 Bad Request: Failed email update when new email is null or empty	1) Create a json user email update request request body where new email is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a POST request to the update email endpoint	UserControllerTest shouldReturnHTTP400 WhenEmailIsNullOr EmptyForEmailUpdate()	1) Status BAD_REQUEST 2) The response body contains relevant error message 3) No interactions with the user service	As expected
TC_52	409 Conflict: Failed to update email when new email already exists	1) Construct a JSON payload for the user email update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw a Duplicate Resource exception 4) Perform a POST request to the update email endpoint	UserControllerTest shouldReturnHTTP409When UpdatingEmailWith ExistingEmail()	1) Status CONFLICT 2) The response body contains relevant error message	As expected
TC_53	3XX Redirection: Successful user redirection to their profile after email update	1) Mock the user service update email to return true 2) Perform a GET request to the update email endpoint	UserControllerTest shouldUpdateEmailAnd RedireUserToTheir ProfileWhenEmailUpdate TokenIsValid()	1) Status 3XXRedirection 2) The Location header contains the redirection url	As expected
TC_54	3XX Redirection: Successful user redirection to email update error page after	1) Mock the user service update email to return true 2) Perform a GET request to the update email	UserControllerTest shouldNotUpdateEmailAnd RedirectUserToEmailUpdate ErrorPageWhenEmail UpdateTokenIsInValid()	1) Status 3XXRedirection 2) The Location header contains the redirection url	As expected

	failed email update	endpoint			
TC_55	404 Not found: When user is not found to update email	1) Construct a JSON payload for the user email update request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw a Resource Not Found exception 3) Perform a POST request to the update email endpoint	UserControllerTest shouldReturnHTTP404 WhenUserIsNotFound ToUpdateEmail ()	1) Status NOT_FOUND 2) The response body contains relevant error message	As expected
TC_56	401 Unauthorized: When unauthenticated user calls update email	1) Construct a JSON payload for the user email update request 2) Perform a POST request to the update email endpoint	UserControllerTest shouldReturnHTTP401When UpdateEmailIsCalledBy UnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_57	400 Bad Request: When an invalid date is provided to get user history in range	1) Construct a JSON payload with the expected response body 3) Mock the user service to throw an Illegal State exception 3) Perform a GET request to the user history endpoint	UserControllerTest shouldReturnHTTP400When AtLeastOneDateIsInvalid ToGetUserHistoryInRange	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_58	401 Unauthorized: When unauthenticated user calls get history	1) Perform a GET request to the user history endpoint	UserControllerTest shouldReturnHTTP401When GetHistoryIsCalledBy UnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected
TC_59	204 No Content: Successful account deletion	1) Construct a JSON payload for the user account delete request 2) Mock the user service 3) Perform a PUT	UserControllerTest shouldReturnHTTP204When AccountIsDeleted()	1) Status NO_CONTENT 2) User service delete account was called once	As expected

		request to the user account endpoint			
TC_60	400 Bad Request: Failed account deletion when password is null or empty	1) Construct a JSON payload for the user account delete request where password is null and empty 2) Construct a JSON payload with the expected response body 3) Perform a PUT request to the user account endpoint	UserControllerTest shouldReturnHTTP400 WhenPasswordIsNullOr Empty ForAccount Deletion()	1) Status BAD_REQUEST 2) The response body contains relevant error message 3) No interactions with the user service	As expected
TC_61	400 Bad Request: Failed account deletion when password is wrong	1) Construct a JSON payload for the user account delete request 2) Construct a JSON payload with the expected response body 3) Mock the user service to throw Bad Credentials exception 4) Perform a PUT request to the user account endpoint	UserControllerTest shouldReturnHTTP400When PasswordIsWrongFor AccountDeletion()	1) Status BAD_REQUEST 2) The response body contains relevant error message	As expected
TC_62	401 Unauthorized: When unauthenticated user calls delete account	1) Construct a JSON payload for the user account delete request 2) Perform a PUT request to the user account endpoint	UserControllerTest shouldReturnHTTP401When DeleteAccountIsCalledBy UnauthenticatedUser()	1) Status UNAUTHORIZED 2) No interactions with the user service	As expected

Integration Tests

Test Case ID	Description	Test Steps	Test Method	Expected Output	Actual Output
TC_01	Successful user login	1) Create a JSON payload for the register request 2)	AuthIT shouldLoginUser()	1) Status CREATED for signup 2) Status OK for login 3) JWT	As expected

		Perform a POST request to the signup endpoint 3) Create a JSON payload for the login request 4) Perform a POST request to the login endpoint		is present in the response body in both cases	
TC_02	Successful password reset	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint 3) Create a JSON payload for the password reset request 4) Perform a POST request to the password reset endpoint 5) Create a JSON payload for the password reset confirmation request with the token from the email link and the new password 6) Perform a PUT request to the password reset confirmation endpoint	AuthIT shouldResetPassword()	1) Status ACCEPTED for the initial request 2) Status NO_CONTENT for the final request 3) User received a link on their email to reset their password 4) User received a confirmation email that the password was reset	As expected
TC_03	Successfully getting user	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint and receive a JWT 3) Perform a GET request to the user endpoint with the token	UserIT shouldGetUser()	1) Status OK 2) Response values match user values	As expected
TC_04	Successfully getting user profile	1) Create a JSON payload for the register request 2)	UserIT shouldGetUserProfile()	1) Status OK 2) Response values match user profile	As expected

		Perform a POST request to the signup endpoint and receive a JWT 3) Perform a GET request to the user profile endpoint with the token		values	
TC_05	Successful user profile update	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint and receive a JWT 3) Create a JSON payload for the user profile update request 4) Perform a PUT request to the user profile endpoint with the token 5) Perform a GET request to the user profile endpoint with the token	UserIT shouldUpdateUserProfile()	1) Status NO_CONTENT for the update request 2) Response values match user profile updated values	As expected
TC_06	Successful user password update	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint and receive a JWT 3) Create a JSON payload for the user password update request 4) Perform a PUT request to the user password endpoint with the token 5) Create a JSON payload for the login request with the updated password 6) Perform a POST	UserIT shouldUpdateUser Password()	1) Status NO_CONTENT for the update request 2) User logs in successfully with the new password	As expected As expected

		request to the login endpoint			
TC_07	Successful user email update	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint and receive a JWT 3) Create a JSON payload for the user email update request 4) Perform a POST request to the user email endpoint with the token (User clicks on the email verification link) 5) Create a JSON payload for the login request with the updated email 6) Perform a POST request to the login endpoint	UserIT shouldUpdateUser Email()	1) Status NO_CONTENT for the update request 2) User logs in successfully with the new email	As expected As expected
TC_08	Successful user account deletion	1) Create a JSON payload for the register request 2) Perform a POST request to the signup endpoint and receive a JWT 3) Create a JSON payload for the user account delete request request 4) Perform a PUT request to the user account endpoint with the token 5) Create a JSON payload for the login request 6) Perform a POST request to the login endpoint	UserIT shouldUpdateUser Email()	1) Status NO_CONTENT for the delete request 2) User logs in results in 401 because account doesn't exist	As expected As expected