Backend Test Cases

Controller

AppointmentControllerTest

Class	Description	Test Step	Expected results
`AppointmentControllerTest`	**Retrieve existing appointment by ID**	• Retrieve an appointment using a valid ID.	Returns the appointment entity matching the provided ID.
	Retrieve non-existent appointment by ID	• Attempt to retrieve an appointment using a non-existent ID.	Returns `null`, indicating no appointment found.
	Retrieve all appointments for a specific user ID	• Retrieve all appointments linked	Returns list of appointments associated

		to a specific patient ID.	with the specified patient ID.
	Create a new appointment successfully	• Create a new appointment with valid patient and provider IDs.	Returns `OK` status with confirmation of new appointment creation.
	Update an existing appointment successfully	• Update an existing appointment.	Returns `OK` status with confirmation of appointment update.
	Fail appointment creation with non-existent patient	• Attempt to create an appointment with a non-existent patient ID.	Returns `BAD_REQUEST` status due to non-existent patient.
	Fail appointment creation with non-existent doctor	• Attempt to create an appointment with a non-existent doctor ID.	Returns `BAD_REQUEST` status due to non-existent doctor.
	Fail appointment creation with invalid fields	• Attempt to create an appointment with invalid fields.	Returns `BAD_REQUEST` status due to invalid appointment fields.
	Update user note successfully	• Update a user note for a specific appointment ID.	Returns `OK` status with confirmation of user note update.
	Fail user note update on non-existent appointment	• Attempt to update a user note on a non-existent appointment ID.	Returns `BAD_REQUEST` status due to non-existent appointment.

AuthControllerTest

Class	Description	Test Step	Expected results
`AuthControllerTest`	**Log in with valid credentials**	• Log in using a valid username and password.	Returns `LoginResponse` with valid JWT token, refresh token, and user ID.
	Log in with invalid credentials	• Log in using an invalid username or password.	Returns `UNAUTHORIZED` status with a message about incorrect credentials.
	Update password successfully	• Update the password using the correct old password.	Returns `OK` status with a confirmation of successful password update.
	Fail password update with incorrect old	• Attempt to update password using the	Returns `BAD_REQUEST` status with an advisory

	password	wrong old password.	message to check credentials.

DoctorControllerTest

Class	Description	Test Step	Expected results
`DoctorControllerTest`	**Retrieve doctor by ID**	- Retrieve a doctor using a valid ID.	Returns the doctor entity matching the provided ID.
	Retrieve non-existent doctor by ID	- Attempt to retrieve a doctor using a non-existent ID.	Returns `null`, indicating no doctor found.
	Retrieve all doctors by patient ID	- Retrieve all doctors associated with a specified patient ID.	Returns a list of doctors linked to the specified patient.
	Retrieve all doctors	- Retrieve all unique doctors.	Returns a unique list of all doctors.
	Create doctor successfully	- Create a doctor from a valid DTO.	Returns `OK` status with confirmation of new doctor creation.
	Fail to create new doctor if invalid	- Attempt to create a doctor from an invalid DTO.	Returns `BAD_REQUEST` status due to missing or invalid doctor ID.
	Delete doctor by ID successfully	- Delete a doctor by ID.	Returns `OK` status with confirmation of doctor deletion.
	Fail to delete doctor with appointments	- Attempt to delete a doctor who still has appointments.	Returns `BAD_REQUEST` status due to existing appointments preventing deletion.
	Fail to delete doctor due to exception	- Attempt to delete a doctor but an exception occurs.	Returns `BAD_REQUEST` status with an error message from the exception.

FacilityControllerTest

Class	Description	Test Step	Expected results
`FacilityControllerTest`	**Retrieve facility by ID when exists**	- Retrieve a facility using a valid ID.	Returns the facility entity matching the provided ID.
	Retrieve facility by ID when does not exist	- Attempt to retrieve a facility using a non-existent ID.	Returns `null`, indicating no facility found.

	Retrieve all facilities sorted by name	• Retrieve all facilities and sort by name.	Returns a list of all facilities, sorted by name.
	Retrieve facilities by type	• Retrieve facilities filtered by a specific type.	Returns a list of facilities filtered by the specified type.
	Create facility successfully	• Create a facility using a valid DTO.	Returns `OK` status with confirmation of new facility creation.
	Delete facility successfully	• Delete a facility by ID.	Returns `OK` status with confirmation of facility deletion.
	Fail to delete facility when not found	• Attempt to delete a facility using a non-existent ID.	Returns `BAD_REQUEST` status with message about facility not found or invalid.
	Fail to delete facility when invalid	• Attempt to delete a facility that fails validation checks.	Returns `BAD_REQUEST` status with message about facility not found or invalid.

FileControllerTest

Class	Description	Test Step	Expected results
`FileControllerTest`	**Successfully upload patients file**	• Upload a file containing patient data.	Returns `OK` status with a message listing the IDs of successfully created users.
	Successfully upload patients file with failures	• Upload a patient file with some entries failing.	Returns `OK` status listing successful and failed user IDs.
	Handle exception during patients upload	• Attempt to upload patient data, but an exception occurs.	Returns `BAD_REQUEST` status with an error message detailing the issue.
	Successfully upload appointments file	• Upload a file containing appointment data.	Returns `OK` status with a message listing the IDs of successfully uploaded appointments.
	Successfully upload appointments file with failures	• Upload an appointment file with some entries failing.	Returns `OK` status listing successful and failed appointment IDs.

	Handle exception during appointments upload	- Attempt to upload appointment data, but an exception occurs.	Returns `BAD_REQUEST` status with an error message detailing the issue.

MessageControllerTest

Class	Description	Test Step	Expected Results
`MessageControllerTest`	createMessage should return created message	- Create a message with a valid request.	Returns `OK` status with a response body indicating message ID.
	getMessagesByPatientId should return messages	- Retrieve messages for a valid patient ID.	Returns a list of `MessageEntity` objects associated with the patient.
	deleteMessage should call service to delete message	- Delete a message by its ID.	Calls the `deleteMessage` service method without returning a value.

NotificationControllerTest

Class	Description	Test Step	Expected Results
`NotificationControllerTest`	sendNotification should return OK response	- Send a notification with a valid request.	Returns `OK` status with the response body "Success".
	sendNotification should handle IOException	- Send a notification but simulate an `IOException`.	Returns `INTERNAL_SERVER_ERROR` status with a message about the failure.
	registerDeviceToken should return OK when patient exists	- Register a device token for an existing patient.	Returns `OK` status with a message "Device token registered successfully".
	registerDeviceToken should return NOT_FOUND when patient does not exist	- Register a device token for a non-existent patient.	Returns `NOT_FOUND` status with a message "Patient not found".
	getDeviceToken should return OK with token when patient exists	- Retrieve a device token for an existing patient.	Returns `OK` status with the device token in the response body.
	getDeviceToken should return NOT_FOUND when patient does not exist	- Retrieve a device token for a non-existent patient.	Returns `NOT_FOUND` status with `null` in the response body.
	getDeviceToken should return NOT_FOUND when patient has no token	- Retrieve a device token for a patient with no registered token.	Returns `NOT_FOUND` status with `null` in the response body.

ResourceControllerTest

Class	Description	Test Step	Expected results
`ResourceControllerTest`	**Retrieve a resource by ID when it exists**	• Retrieve a resource using a valid ID.	Returns the resource entity matching the provided ID.
	Retrieve a resource by ID when it does not exist	• Attempt to retrieve a resource using a non-existent ID.	Returns `null`, indicating no resource found.
	Retrieve all resources for a user	• Retrieve all resources associated with a specified user ID.	Returns a list of resources linked to the specified user.
	Create a new resource successfully	• Create a resource using a valid DTO.	Returns `OK` status with a message confirming the resource creation and listing affected patient IDs.
	Create a new resource, handle missing patient	• Create a resource excluding a patient who does not exist.	Returns `OK` status with a message listing successfully linked patient IDs.
	Fail to create a resource due to invalid data	• Attempt to create a resource with invalid or missing data.	Returns `BAD_REQUEST` status with a message indicating failure due to invalid data.
	Delete a resource successfully	• Delete a resource by its ID.	Returns `OK` status with a message confirming the resource deletion.
	Fail to delete a resource due to exception	• Attempt to delete a resource but encounter an exception.	Returns `BAD_REQUEST` status with a detailed error message.

UserControllerTest

Class	Description	Test Step	Expected results
`UserControllerTest`	**Retrieve a patient by ID**	• Retrieve a patient using a valid patient ID.	Returns the patient entity matching the provided patient ID.
	Retrieve all patients	• Retrieve all patient entities.	Returns a list of all patient entities.
	Retrieve non-existent patient by ID	• Attempt to retrieve a patient using an invalid ID.	Returns `null`, indicating no patient found.

	Create patient successfully	- Create a patient from a valid DTO.	Returns `OK` status with a message confirming patient creation or update.
	Fail to create patient with invalid data	- Attempt to create a patient with invalid data.	Returns `BAD_REQUEST` status with a message indicating missing or invalid ID.
	Retrieve an admin by ID	- Retrieve an admin using a valid MMS ID.	Returns the admin entity matching the provided MMS ID.
	Retrieve non-existent admin by ID	- Attempt to retrieve an admin using an invalid ID.	Returns `null`, indicating no admin found.
	Create admin successfully	- Create an admin from a valid DTO.	Returns `OK` status with a message confirming admin creation or update.

Mapping

AppointmentMapperTest

Class	Description	Test Step	Expected results
`AppointmentMapperTest`	**Map AppointmentDTO to AppointmentEntity**	- Map a fully populated AppointmentDTO to an AppointmentEntity.	The mapped entity matches the expected AppointmentEntity based on the provided DTO.
	Update existing AppointmentEntity	- Update an existing AppointmentEntity with new data from another entity.	The updated entity reflects changes and matches the expected output.
	Map an appointment from HTML format	- Map appointment data from HTML (simulated) to an AppointmentEntity.	Successfully maps HTML data to an AppointmentEntity, with a status of `SUCCESS`.
	Throw exception if missing required column	- Attempt to map appointment from HTML with missing required columns.	Throws `ColumnError` due to missing necessary data for mapping.
	Throw exception if missing required ID column	- Attempt to map appointment from HTML with missing ID column.	Throws `ColumnError` due to missing the ID column, which is necessary for mapping.

	Throw ID exception if missing required ID value	• Attempt to map appointment from HTML with missing ID value.	Throws `IdException` due to missing ID value, critical for mapping.
	Throw ID exception if ID value is non-integer	• Attempt to map appointment from HTML where ID value is not an integer.	Throws `IdException` due to non-integer ID value, which is necessary for correct mapping.
	Return failed status if an appointment has missing value	• Map appointment from HTML with missing values.	Returns a failed status because the missing values lead to incomplete appointment data.

DoctorMapperTest

Class	Description	Test Step	Expected Results
`DoctorMapperTest`	**Map a DoctorDTO to DoctorEntity**	• Map a fully populated DoctorDTO to a DoctorEntity.	The mapped entity matches the expected DoctorEntity based on the provided DTO.

ResourceMapperTest

Class	Description	Test Step	Expected Results
`ResourceMapperTest`	**Map Resource DTO to User Entity**	• Map a fully populated ResourceDTO to a ResourceEntity.	The mapped entity's text and link match those in the DTO, and the entity ID is not null.

UserMapperTest

Class	Description	Test Step	Expected Results
`UserMapperTest`	**Map a patient DTO to user entity**	• Map a fully populated PatientDTO to a PatientEntity.	The mapped entity matches the expected PatientEntity based on the provided DTO.
	Update existing patient	• Update an existing PatientEntity with new data.	The updated entity reflects the correct data and matches the expected output.
	Map a patient from HTML	• Map patient data from HTML to a PatientEntity.	Successfully maps HTML data to a

			PatientEntity, with a status of `SUCCESS`.
	Map a patient from HTML even if optional column is missing	• Map patient data from HTML with a missing optional column.	Successfully maps HTML data to a PatientEntity despite missing optional column.
	Throw exception if data array is mismatch	• Attempt to map patient from HTML with mismatched data array.	Throws an exception due to mismatch in data array configuration.
	Throw exception if data array has missing required column	• Attempt to map patient from HTML with missing required column.	Throws `ColumnError` due to the absence of a required column.
	Throw exception if data array has missing required id column	• Attempt to map patient from HTML with missing ID column.	Throws `ColumnError` due to missing ID column, necessary for mapping.
	Return failure status if data array has missing required value	• Map patient from HTML with missing values.	Returns a failure status because the missing values lead to incomplete patient data.
	Throw exception if data array has missing required id	• Attempt to map patient from HTML with missing ID.	Throws `IdException` due to missing patient ID, critical for mapping.
	Throw exception if data array has non-integer id	• Attempt to map patient from HTML where ID value is not an integer.	Throws `IdException` due to non-integer ID value, which is necessary for correct mapping.
	Map Admin DTO to User Entity	• Map a fully populated AdminDTO to an AdminEntity.	The mapped entity matches the expected AdminEntity based on the provided DTO.

Service

Security

AuthServiceTest

Class	Description	Test Step	Expected Results
`AuthServiceTest`	**Successfully authenticate an email and password**	• Authenticate using a valid email and password.	Returns a triple containing valid JWT, refresh token, and MMS

			ID, confirming successful authentication.
	Fail to authenticate when authentication not valid	• Attempt to authenticate using an incorrect email or password.	Returns `null`, indicating authentication failure due to invalid credentials.

CustomUserDetailsServiceTest

Class	Description	Test Step	Expected Results
`CustomUserDetailsServiceTest`	**Load a patient details by username**	• Load user details by a patient's username.	Returns user details including username and password, with the authority "ROLE_PATIENT".
	Load an admin details by username	• Load user details by an admin's username.	Returns user details including username and password, with the authority "ROLE_ADMIN".

OneTimePasscodeAuthService

Class	Description	Test Step	Expected Results
`OneTimePasscodeAuthServiceTest`	Successfully authenticate one time password	• Authenticate with a valid one-time passcode and password.	Returns a pair `(true, patient)`.
	Fail to authenticate one time password when passcode is wrong	• Attempt to authenticate with an incorrect one-time passcode.	Returns a pair `(false, null)`.
	Fail to authenticate one time password when patient cannot be found	• Attempt to authenticate with a valid passcode but non-existent patient email.	Returns a pair `(false, null)`.

PasscodeServiceTest

Class	Description	Test Step	Expected Results
`PasscodeServiceTest`	Generate a passcode	• Generate a random integer passcode.	Asserts that the generated passcode has a length of 6.

TokenServiceTest

Class	Description	Test Step	Expected Results
`TokenServiceTest`	**Generate a JWT**	• Generate a JWT with specific claims and expiry date.	Asserts that the generated JWT starts with a specific string,

			indicating successful creation.

UsernameServiceTest

Class	Description	Test Step	Expected Results
`UsernameServiceTest`	Generate a username from unseen first name of acceptable length	• Generate a username from a first name of acceptable length.	Returns the first name as the username.
	Generate a username where first name is too long	• Generate a username where the first name exceeds acceptable length.	Returns the last name as the username.
	Generate a username where first name is too short	• Generate a username where the first name is too short.	Returns the last name as the username.
	Generate a username where both first and last name are too short	• Generate a username where both names are too short.	Returns a concatenation of both names.
	Generate a username where both first and last name are too long	• Generate a username where both names exceed acceptable lengths.	Returns a shortened version based on predefined rules.
	Generate a username where first name is too short and last name is too long	• Generate a username with a short first name and a long last name.	Returns a username with a user tag prefix.
	Generate a username where last name is too short and first name is too long	• Generate a username with a long first name and a short last name.	Returns a shortened version of the first name.
	Generate a username where multiple names are taken	• Generate a username in a scenario where several preferred options are taken.	Returns a unique username with a numeric suffix.
	Generate a username from a name with whitespace	• Generate a username from a name containing extra spaces.	Returns a cleaned and formatted username.
	Lowercase entire username & remove whitespace simultaneously	• Generate a username from a name with irregular spacing and capitalization.	Returns a lowercase and concatenated username.
	Pad very short names with user tag	• Generate a username from very short names using a user tag.	Returns a username with a "patient_" prefix.
	Handle null surname	• Generate a username when the surname provided is null.	Returns a username with a user tag prefix.

AppointmentReminderServiceTest

Class	Description	Test Step	Expected Results
`AppointmentReminderServiceTest`	send reminders successfully	- Send reminders for appointments with active patient accounts and valid device tokens.	Sends reminders and saves the appointment entities.
	do not send reminders if patient account is not active	- Attempt to send reminders for appointments where the patient's account is not active.	No reminders sent, no appointment entities saved.
	do not send reminders if patient device token is null	- Attempt to send reminders for appointments where the patient has no device token.	No reminders sent, no appointment entities saved.
	do not send reminders if last notified time is not null	- Attempt to send reminders for appointments already notified.	No reminders sent, no appointment entities saved.

EmailServiceTest

Class	Description	Test Step	Expected Results
`EmailServiceTest`	Successfully send sign up email	- Send an activation or recovery email for an existing patient.	Sends an email using the email sender.
	Throw exception on sign up if patient doesn't exist	- Attempt to send an activation or recovery email for a non-existent patient.	Throws an `Exception`.

FileServiceTest

Class	Description	Test Step	Expected Results
`FileServiceTest`	**Successfully read and upload a user file**	- Upload user data from a valid HTML file.	Successfully uploads user data, returning a pair of empty failed list and list of successful IDs.
	Fail to upload a user file with a missing column	- Attempt to upload user data from a file with missing columns.	Throws `ColumnError` due to missing required columns in the file.
	Successfully read and upload a user file but some user files are failed to be uploaded	- Upload user data with some missing required values.	Successfully uploads some user data while returning a list of IDs that failed to upload.
	Successfully read and upload a user file even	- Upload user data from a file missing	Successfully uploads user data, ignoring

	with missing optional col	optional columns.	missing optional columns.
	Fail to upload corrupted patient data	- Attempt to upload corrupted user data.	Throws an exception due to corruption in the user data file.
	Successfully read and upload an appointment file	- Upload appointment data from a valid HTML file.	Successfully uploads appointment data, returning a pair of empty failed list and list of successful IDs.
	Fail to upload an appointment file with a missing column	- Attempt to upload appointment data from a file with missing columns.	Throws `ColumnError` due to missing required columns in the file.
	Successfully read and upload an appointment file but some appointment files are failed to be uploaded	- Upload appointment data with some missing required values.	Successfully uploads some appointment data while returning a list of IDs that failed to upload.
	Do not upload appointments for which linked patient doesn't exist	- Attempt to upload appointment data linked to non-existent patients.	Skips uploading appointments where the linked patient does not exist.
	Do not upload appointments for which linked doctor doesn't exist	- Attempt to upload appointment data linked to non-existent doctors.	Skips uploading appointments where the linked doctor does not exist.

MessageServiceTest

Class	Description	Test Step	Expected Results
`MessageServiceTest`	createMessage should save and return a new message	- Create a new message and save it to the repository.	Saves the message and returns the saved `MessageEntity`.
	getMessagesByPatientId should return messages for a patient	- Retrieve messages for a specific patient ID.	Returns a list of `MessageEntity` objects associated with the patient.
	deleteMessage should delete a message by message ID	- Delete a message by its ID.	Deletes the message from the repository.

NotificationServiceTest

Class	Description	Test Step	Expected Results
`NotificationServiceTest`	send a message to a token	- Send a notification to a valid device token.	Returns the JSON response indicating success.

	fail send a message because of faulty request	• Attempt to send a notification with an invalid request.	Throws an `IOException`.
	test sending message to token with unsuccessful response	• Send a notification to a device token and receive an unsuccessful response.	Throws an `IOException` with the message "Unexpected code Response{}".

MmsBackendApplication

Class	Description	Test Step	Expected Results
`MmsBackendApplicationTests`	**Context Loads**	• Verify that the application context loads successfully and that the `FacilityController` is properly autowired.	Asserts that the `FacilityController` is not null, indicating successful loading of the application context.