Continuing from the Assignment 2, let's add error handling and exception handling functionality as well as versioning to the project.

1. Error Handling:
    - Create a custom error model to represent API error responses. Include properties like `ErrorCode` and `ErrorMessage`.
    - Implement global error handling by configuring the `ExceptionHandler` middleware in the `Startup` class. Customize the error response format and include the error model.
    - Handle specific exceptions by creating custom exception filters or exception middleware to provide meaningful error messages for different scenarios.
2. Exception Handling:
    - Create custom exceptions for specific error cases related to the business logic of the application.
    - Implement exception handling middleware in the `Startup` class to catch and handle these custom exceptions. Return appropriate error responses with the custom error model created earlier.
3. Versioning:
    - Install the `Microsoft.AspNetCore.Mvc.Versioning` package to enable API versioning in the project.
    - Configure API versioning, specifying the versioning scheme (e.g., URL-based, header-based).
    - Decorate the controller classes with appropriate versioning attributes, such as `[ApiVersion]` or `[ApiVersion("1.0")]`.
    - Create separate controller versions for different API versions if required, using the `[ApiVersion]` attribute.
    - Test the versioning functionality by making requests to different API versions and verifying the responses.
4. Update the existing controllers and actions to handle exceptions and return appropriate error responses.
    - Add exception handling logic within the actions to catch any potential exceptions and return corresponding error responses.
    - Use the custom exception filters or middleware created earlier to handle specific exceptions and transform them into meaningful error messages.
5. Test the error handling and exception handling functionality using tools like Postman or Swagger. Verify that when an error or exception

occurs, the API returns the expected error response with the appropriate status code and message.

Note: Ensure that the error and exception handling logic is consistent across all controllers and actions, and follow best practices for error handling and exception handling in ASP.NET Core.