TestResultEntity testResultEntity = this.findById(testResultFinish.getId());  
TestEntity test = testResultEntity.getTest();  
List<QuestionEntity> listQuestion = test.getQuestions();  
List<AnswerSelectedRequest> listAnswerSelected = testResultFinish.getAnswerSelecteds();  
List<QuestionResultEntity> listQuestionResult = new ArrayList<>();  
int score = 0;  
for (QuestionEntity questionEntity : listQuestion) {  
 List<String> answerSelectedRequest = listAnswerSelected.stream()  
 .filter(question -> questionEntity.getId().equals(question.getQuestionId()))  
 .findFirst()  
 .map(AnswerSelectedRequest::getAnswerIds)  
 .orElse(null);  
 QuestionResultEntity questionResult = new QuestionResultEntity();  
 questionResult.setQuestion(questionEntity);  
 questionResult.setTestResult(testResultEntity);  
 List<AnswerQuestionResultEntity> listAnswerQuestionResult = new ArrayList<>();  
 if (answerSelectedRequest == null || answerSelectedRequest.isEmpty()) {  
 questionResult.setAnswers(listAnswerQuestionResult);  
 questionResult.setStatus(AnswerSelectedStatus.*NOT\_ANSWERED*);  
 } else {  
 boolean isSuccess = true;  
 List<AnswerQuestionResultEntity> listAnswerForQuestionCurrent = new ArrayList<>();  
 for (String answerId : answerSelectedRequest) {  
 AnswerEntity answer = answerRepository.findById(answerId)  
 .orElseThrow(() -> new DataInvalidException(ExceptionVariable.*ANSWER\_NOT\_FOUND*));  
 if (!answer.getIsCorrect()) isSuccess = false;  
 listAnswerForQuestionCurrent.add(new AnswerQuestionResultEntity(answer, questionResult));  
 }  
 questionResult.setAnswers(listAnswerForQuestionCurrent);  
 if (isSuccess) {  
 score++;  
 questionResult.setStatus(AnswerSelectedStatus.*CORRECT*);  
 } else {  
 questionResult.setStatus(AnswerSelectedStatus.*INCORRECT*);  
 }  
 }  
 listQuestionResult.add(questionResult);  
}  
testResultEntity.setScore(score);  
testResultEntity.setFinishDate(testResultFinish.getFinishDate());  
testResultEntity.getAnswerSelecteds().addAll(listQuestionResult);  
testResultEntity.setStatus(TestResultStatus.*COMPLETE*);  
return testResultRepository.save(testResultEntity);