PACEMAKER

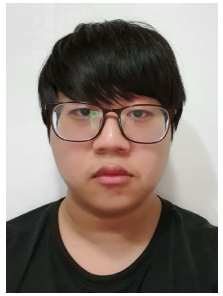
SW Design Specification

곽신우 (201411260)

문성찬 (201511260)

배율희 (201511266)

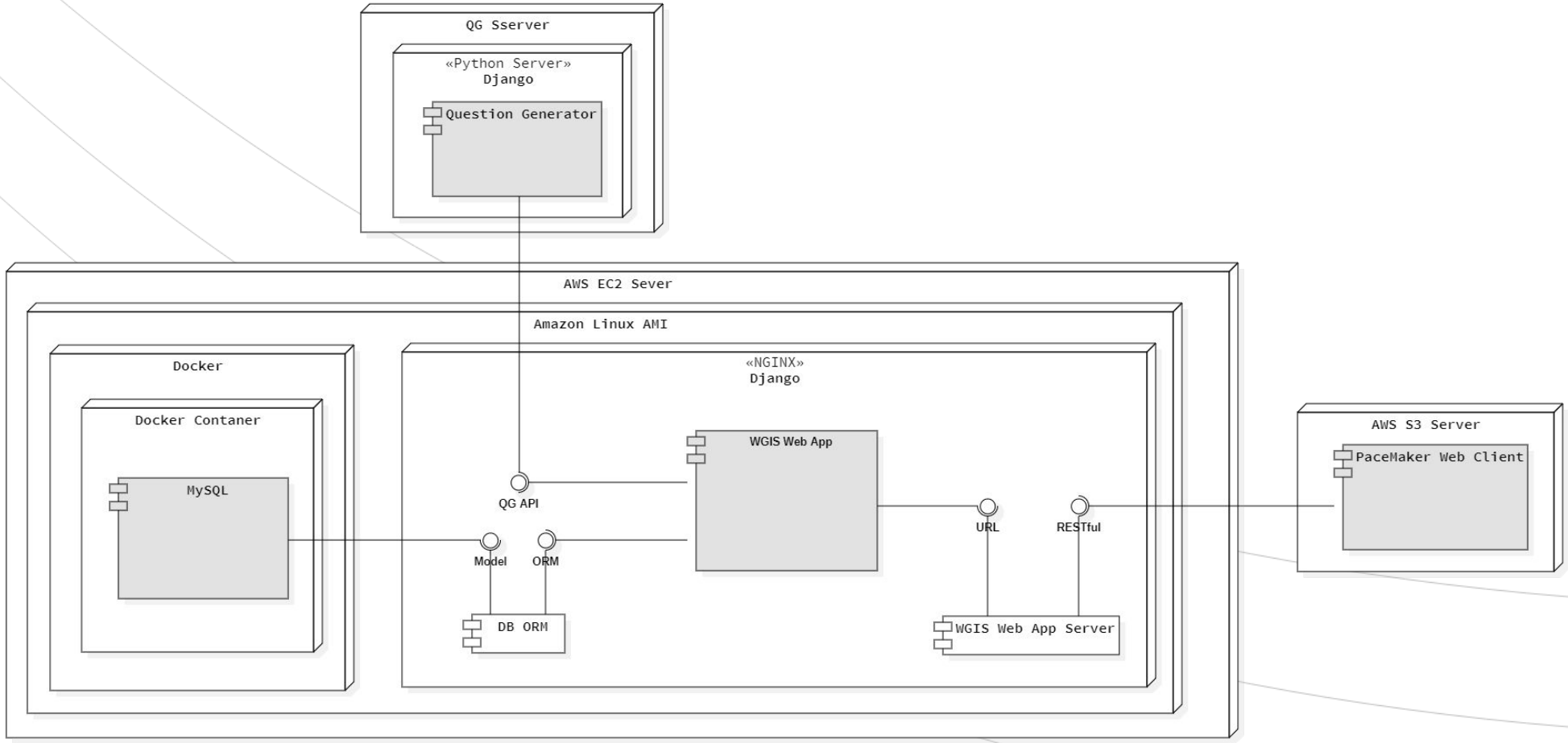
주재빈 (201511298)



CONTENTS

ARCHITECTURE

Deployment Diagram



SUB SYSTEM INTERFACE

Web Client → App Server

No.	Category	Description	No.	Category	Description
WC.1	SignIn	사용자의 이메일과 비밀번호를 받아서 로그인 가능 여부를 판단.	WC.14	EditUnit	자신이 리더인 채널에 존재하는 유닛의 정보를 수정
WC.2	SignUp	시스템에 존재하지 않는 이메일을 새로 등록	WC.15	DeleteUnit	자신이 리더인 채널에 존재하는 유닛을 삭제
WC.3	FindAccount	시스템에 존재하는 이메일에 대해 해당 이메일의 비밀번호를 초기화	WC.16	CreateBKD	새로운 빈 BKD 생성
WC.4	ModifyAccount	로그인이 완료된 계정에 대해, 계정의 정보를 수정	WC.17	EditBKD	존재하는 BKD의 내용을 수정
WC.5	CreateChannel	자신을 리더로 하는 새로운 빈 채널을 생성	WC.18	DeleteBKD	존재하는 BKD 삭제
WC.6	DeleteChannel	자신이 리더인 채널을 삭제	WC.19	RequestBKD	존재하는 BKD의 내용을 요청
WC.7	EditChannel	자신이 리더인 채널 수정	WC.20	RequestQG	BKD를 기반으로 문제 생성을 요청
WC.8	EnterChannel	존재하는 채널에 러너로서 가입	WC.21	VerifyQuestion	생성된 문제에서 사용할 수 없는 문제를 골라서 삭제 요청
WC.9	RequestChannel	가입한 채널의 정보를 요청하여 그 정보를 보여줌	WC.22	MakeReservation	생성된 문제를 기반으로 러너에게 시험지 배포하는 것을 예약
WC.10	ExitChannel	러너로서 가입한 채널에서 퇴장	WC.23	RequestBoard	채널의 정보를 새로 요청
WC.11	Refresh	클라이언트 뷰의 새로고침에 필요한 데이터를 요청 (시험지 요청 포함)	WC.24	RequestBKDs	문서목록의 정보를 새로 요청
WC.12	RequestUnit	유닛 ID를 이용하여 유닛의 정보를 요청	WC.25	RequestPaper	배포된 시험지의 ID를 이용하여 시험지 데이터 요청
WC.13	CreateUnit	자신이 리더인 채널에 새로운 빈 유닛을 추가함	WC.26	SubmitPaper	풀이가 완료된 시험지를 서버로 전송

SUB SYSTEM INTERFACE

App Server → QG Server

<table>
<tr><th rowspan="2">No.</th><th rowspan="2">Category</th><th colspan="2">API</th><th rowspan="2">Description</th></tr>
<tr><th>Type</th><th>REST HTTP POST Request</th></tr>
<tr><td rowspan="6">QG.1</td><td rowspan="6">qg_per_bkd
(App Server - QG Server)</td><td>Destination</td><td>http://117.16.136.170/restful/qg</td><td rowspan="6">Payload:
- JSON 형태 key-value 문자열
- key: "bkd"
- value: BKD 내용을 담은 문자열

Response:
- JSON 형태 key-value 문자열

> key: "passages"
> value: BKD 내용에서 추출한 단락과
해당 단락으로부터 생성된 질문-정답 쌍들

>> key: "text"
>> value: 단락의 내용을 담은 문자열

>> key: "nouns"
>> value: 단락에서 추출되어 TF-IDF 지표로 내림차순 정렬된 단어 리스트

>> key: "aqset"
>> value: 상위 6단어를 정답으로 하여 자동생성한 정답-질문 쌍 리스트</td></tr>
<tr><td>Payload</td><td>JSON 포맷 문자열</td></tr>
<tr><td colspan="2">{
"bkd": "text of bkd"
}</td></tr>
<tr><td>Response</td><td>JSON 포맷 문자열</td></tr>
<tr><td colspan="2">{
"passages": [
{
"text": "text of bkd",
"nouns": [
"text",
"bkd"
],
"aqset": [
[
"text",
"What is the name of the text that bkd is a part of ?"
],
[
"bkd",
"What is the text of the document that describes the contents of the document ?"
]
]
}
]
}</td></tr>
</table>

SUB SYSTEM INTERFACE

App Server → QG Server

No.	Category	API		Description
		Type	**TCP Socket**	
QG.2	qg_per_paset QG Server - UniLMv1 Socket Server	**Destination**	http://117.16.136.170:2593	Payload: - BKD 내용과 정답으로 사용할 명사구를 [SEP] 토큰으로 구분한 문자열 - 이를 여럿으로 묶을 때 '\n' 으로 구분한 문자열의 바이트 배열 Response: - 각 정답에 대하여 생성된 문장들을 '\n'으로 구분하여 묶어낸 문자열의 바이트 배열
		Payload	10KB 이내의 ASCII 문자열	
		b"text of bkd [SEP] text\n text of bkd [SEP] bkd\n."		
		Response	4KB 이내의 ASCII 바이트 배열	
		b"What is the name of the text that bkd is a part of ?\n What is the text of the document that describes the contents of the document ?"		

SUB SYSTEM INTERFACE

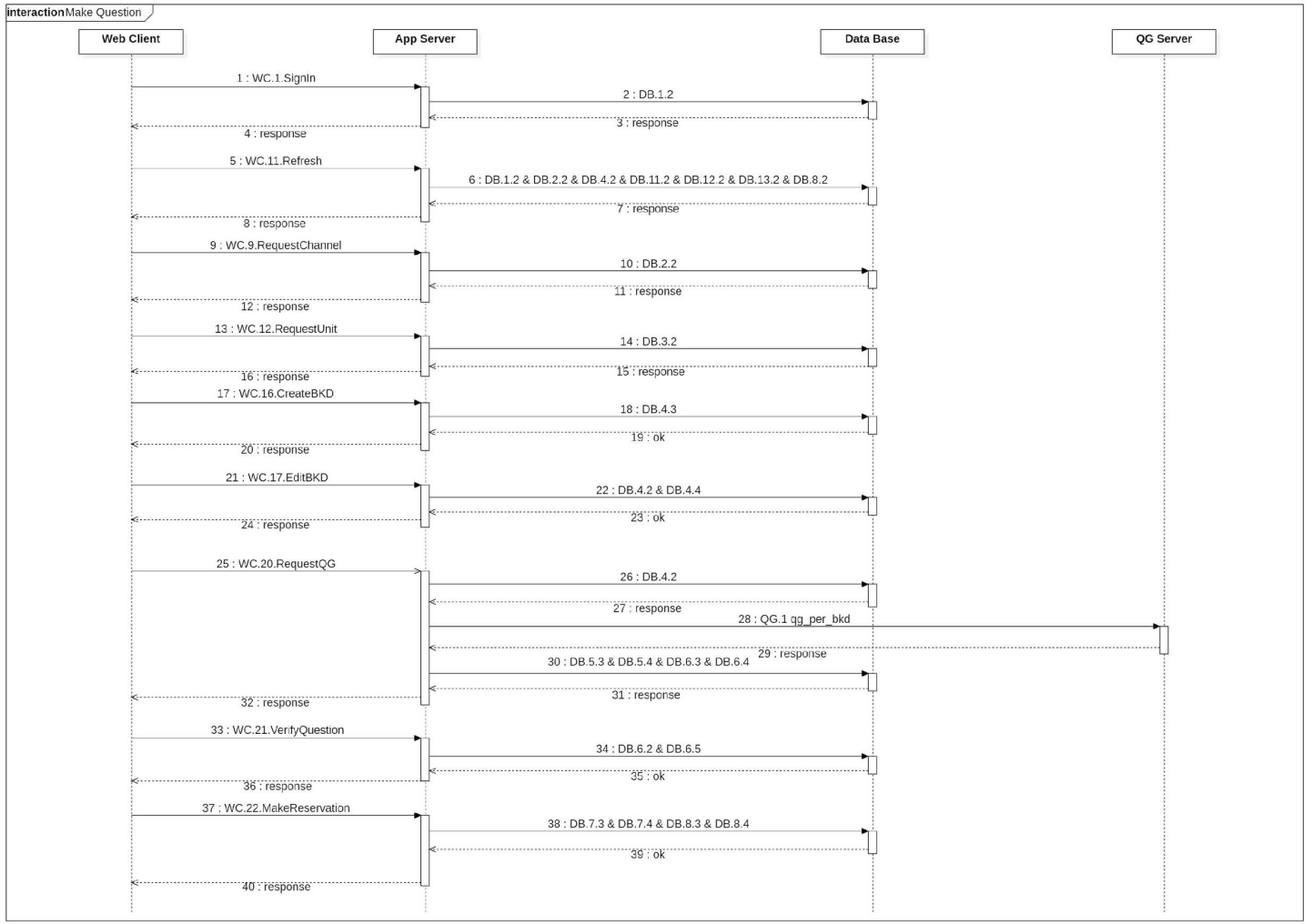
App Server → Data Base

No.		Category	API Type	API Django ORM	Description
DB.1	DB.1.1	User	User.objects.filter()		존재 여부 반환
	DB.1.2		User.objects.get()		객체 반환
	DB.1.3		User.objects.create()		객체 생성
	DB.1.4		User.save()		객체 저장
	DB.1.5		User.delete()		객체 삭제
DB.2	DB.2.1	Channel	Channel.objects.filter()		존재 여부 반환
	DB.2.2		Channel.objects.get()		객체 반환
	DB.2.3		Channel.objects.create()		객체 생성
	DB.2.4		Channel.save()		객체 저장
	DB.2.5		Channel.delete()		객체 삭제
DB.3	DB.3.1	Unit	Unit.objects.filter()		존재 여부 반환
	DB.3.2		Unit.objects.get()		객체 반환
	DB.3.3		Unit.objects.create()		객체 생성
	DB.3.4		Unit.save()		객체 저장
	DB.3.5		Unit.delete()		객체 삭제
DB.4	DB.4.1	BKD	BKD.objects.filter()		존재 여부 반환
	DB.4.2		BKD.objects.get()		객체 반환
	DB.4.3		BKD.objects.create()		객체 생성
	DB.4.4		BKD.save()		객체 저장
	DB.4.5		BKD.delete()		객체 삭제
DB.5	DB.5.1	QASet	QASet.objects.filter()		존재 여부 반환
	DB.5.2		QASet.objects.get()		객체 반환
	DB.5.3		QASet.objects.create()		객체 생성
	DB.5.4		QASet.save()		객체 저장
	DB.5.5		QASet.delete()		객체 삭제
DB.6	DB.6.1	QAPair	QAPair.objects.filter()		존재 여부 반환
	DB.6.2		QAPair.objects.get()		객체 반환
	DB.6.3		QAPair.objects.create()		객체 생성
	DB.6.4		QAPair.save()		객체 저장
	DB.6.5		QAPair.delete()		객체 삭제
DB.7	DB.7.1	TestPlan	TestPlan.objects.filter()		존재 여부 반환
	DB.7.2		TestPlan.objects.get()		객체 반환
	DB.7.3		TestPlan.objects.create()		객체 생성
	DB.7.4		TestPlan.save()		객체 저장
	DB.7.5		TestPlan.delete()		객체 삭제
DB.8	DB.8.1	TestSet	TestSet.objects.filter()		존재 여부 반환
	DB.8.2		TestSet.objects.get()		객체 반환
	DB.8.3		TestSet.objects.create()		객체 생성
	DB.8.4		TestSet.save()		객체 저장
	DB.8.5		TestSet.delete()		객체 삭제
DB.9	DB.9.1	TestPair	TestPair.objects.filter()		존재 여부 반환
	DB.9.2		TestPair.objects.get()		객체 반환
	DB.9.3		TestPair.objects.create()		객체 생성
	DB.9.4		TestPair.save()		객체 저장
	DB.9.5		TestPair.delete()		객체 삭제

No.		Category	API Type	API Django ORM	Description
DB.10	DB.10.1	News	News.objects.filter()		존재 여부 반환
	DB.10.2		News.objects.get()		객체 반환
	DB.10.3		News.objects.create()		객체 생성
	DB.10.4		News.save()		객체 저장
	DB.10.5		News.delete()		객체 삭제
DB.11	DB.11.1	Guest	Guest.objects.filter()		존재 여부 반환
	DB.11.2		Guest.objects.get()		객체 반환
	DB.11.3		Guest.objects.create()		객체 생성
	DB.11.4		Guest.save()		객체 저장
	DB.11.5		Guest.delete()		객체 삭제
DB.12	DB.12.1	Host	Host.objects.filter()		존재 여부 반환
	DB.12.2		Host.objects.get()		객체 반환
	DB.12.3		Host.objects.create()		객체 생성
	DB.12.4		Host.save()		객체 저장
	DB.12.5		Host.delete()		객체 삭제
DB.13	DB.13.1	BKDOwner	BKDOwner.objects.filter()		존재 여부 반환
	DB.13.2		BKDOwner.objects.get()		객체 반환
	DB.13.3		BKDOwner.objects.create()		객체 생성
	DB.13.4		BKDOwner.save()		객체 저장
	DB.13.5		BKDOwner.delete()		객체 삭제
DB.14	DB.14.1	BKDToQA	BKDToQA.objects.filter()		존재 여부 반환
	DB.14.2		BKDToQA.objects.get()		객체 반환
	DB.14.3		BKDToQA.objects.create()		객체 생성
	DB.14.4		BKDToQA.save()		객체 저장
	DB.14.5		BKDToQA.delete()		객체 삭제
DB.15	DB.15.1	UnitQA	UnitQA.objects.filter()		존재 여부 반환
	DB.15.2		UnitQA.objects.get()		객체 반환
	DB.15.3		UnitQA.objects.create()		객체 생성
	DB.15.4		UnitQA.save()		객체 저장
	DB.15.5		UnitQA.delete()		객체 삭제
DB.16	DB.16.1	UnitTest	UnitTest.objects.filter()		존재 여부 반환
	DB.16.2		UnitTest.objects.get()		객체 반환
	DB.16.3		UnitTest.objects.create()		객체 생성
	DB.16.4		UnitTest.save()		객체 저장
	DB.16.5		UnitTest.delete()		객체 삭제
DB.17	DB.17.1	UserNews	UserNews.objects.filter()		존재 여부 반환
	DB.17.2		UserNews.objects.get()		객체 반환
	DB.17.3		UserNews.objects.create()		객체 생성
	DB.17.4		UserNews.save()		객체 저장
	DB.17.5		UserNews.delete()		객체 삭제

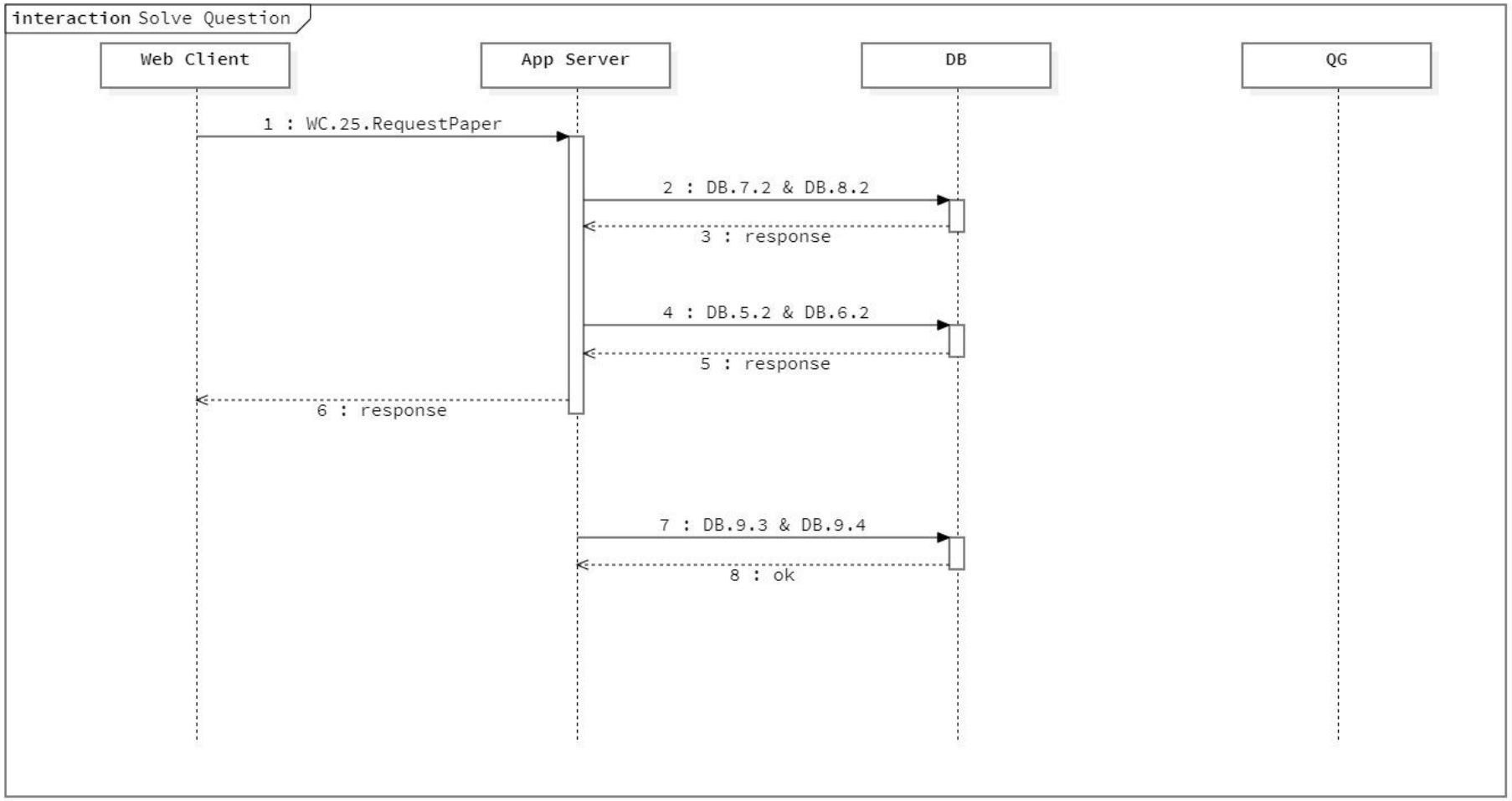
PRIMARY SCENARIO

Leader - Make Question



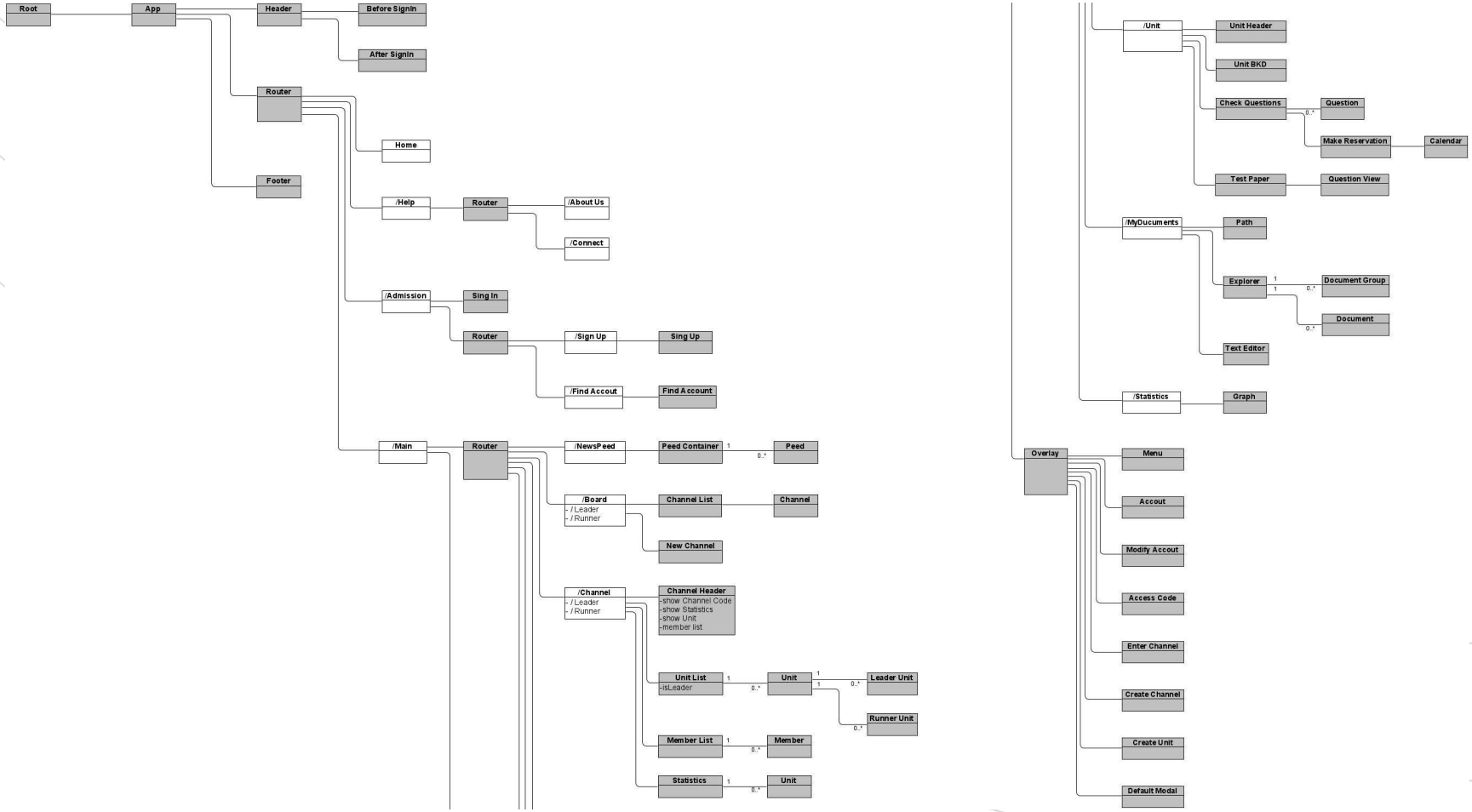
PRIMARY SCENARIO

Runner - Solve Question



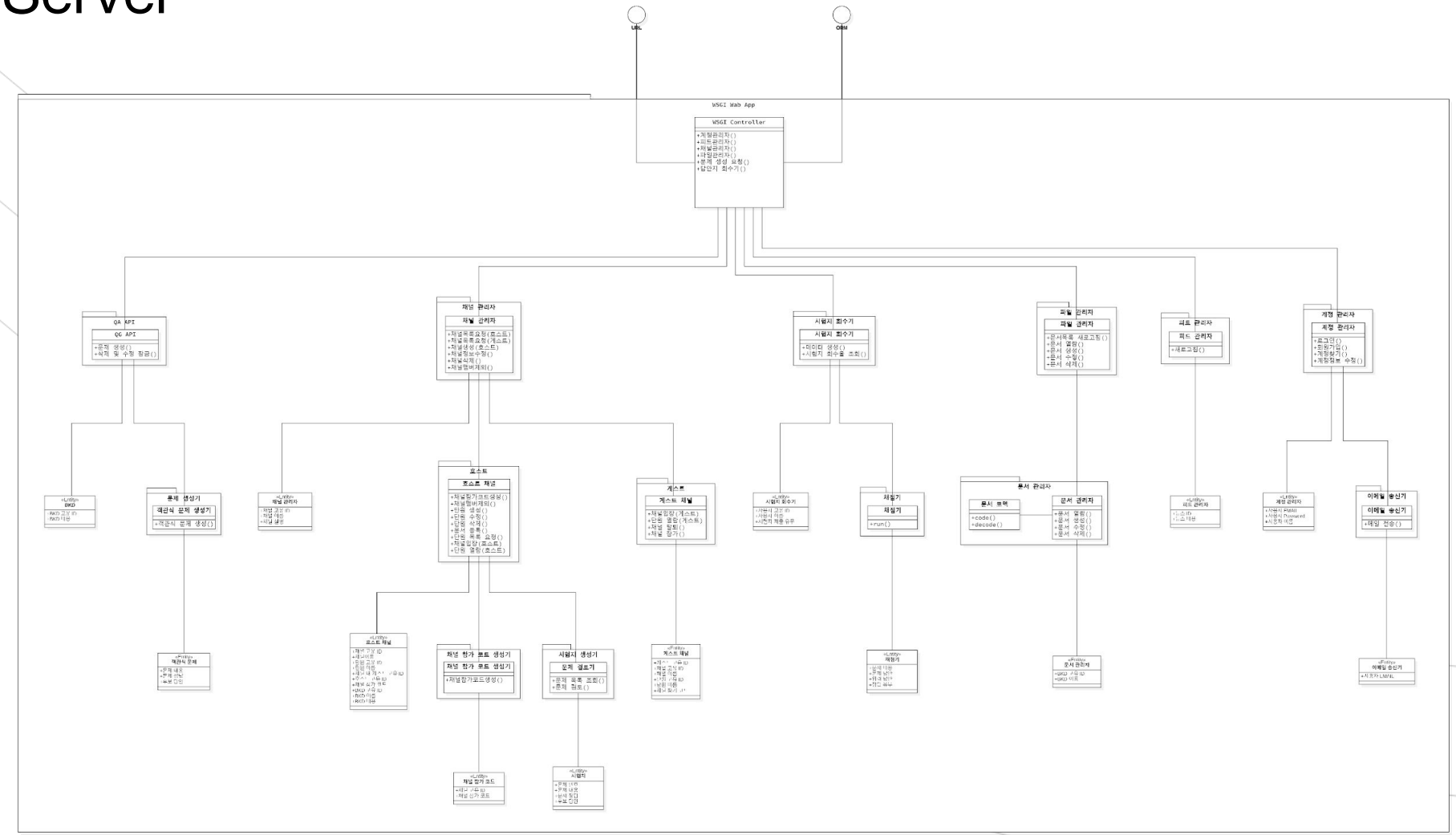
SUB SYSTEM DESCRIPTION

Web Client



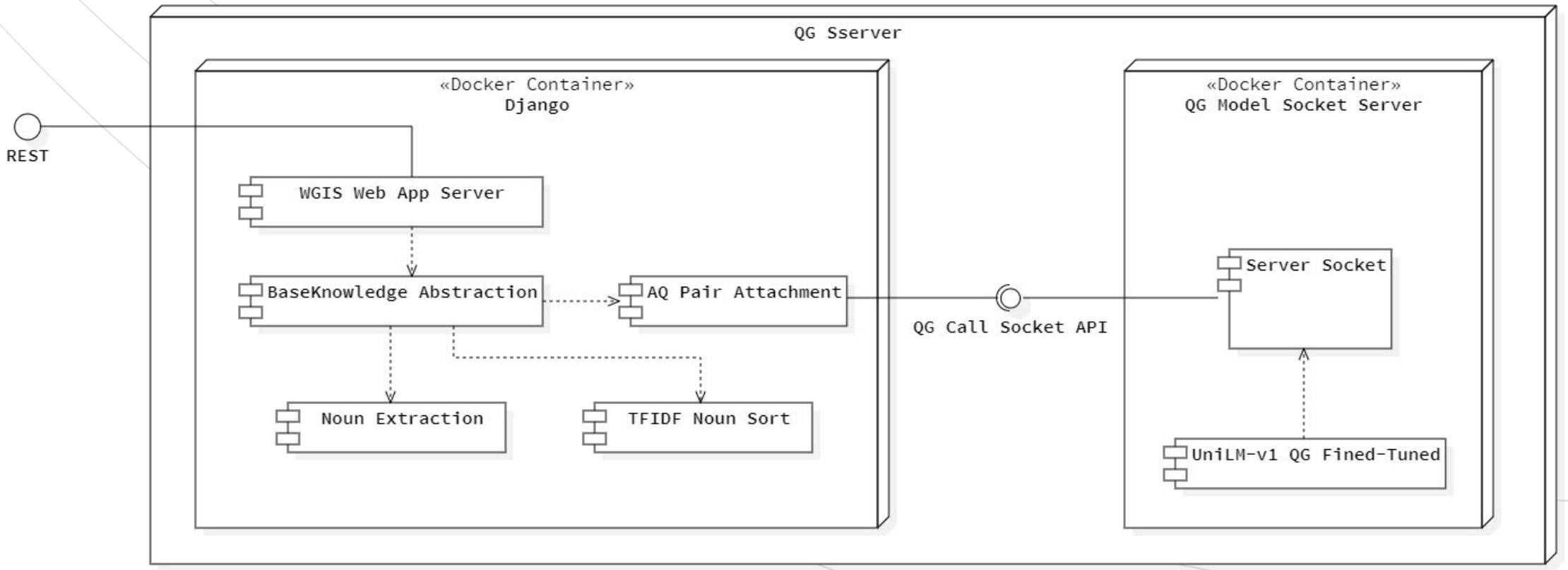
SUB SYSTEM DESCRIPTION

App Server



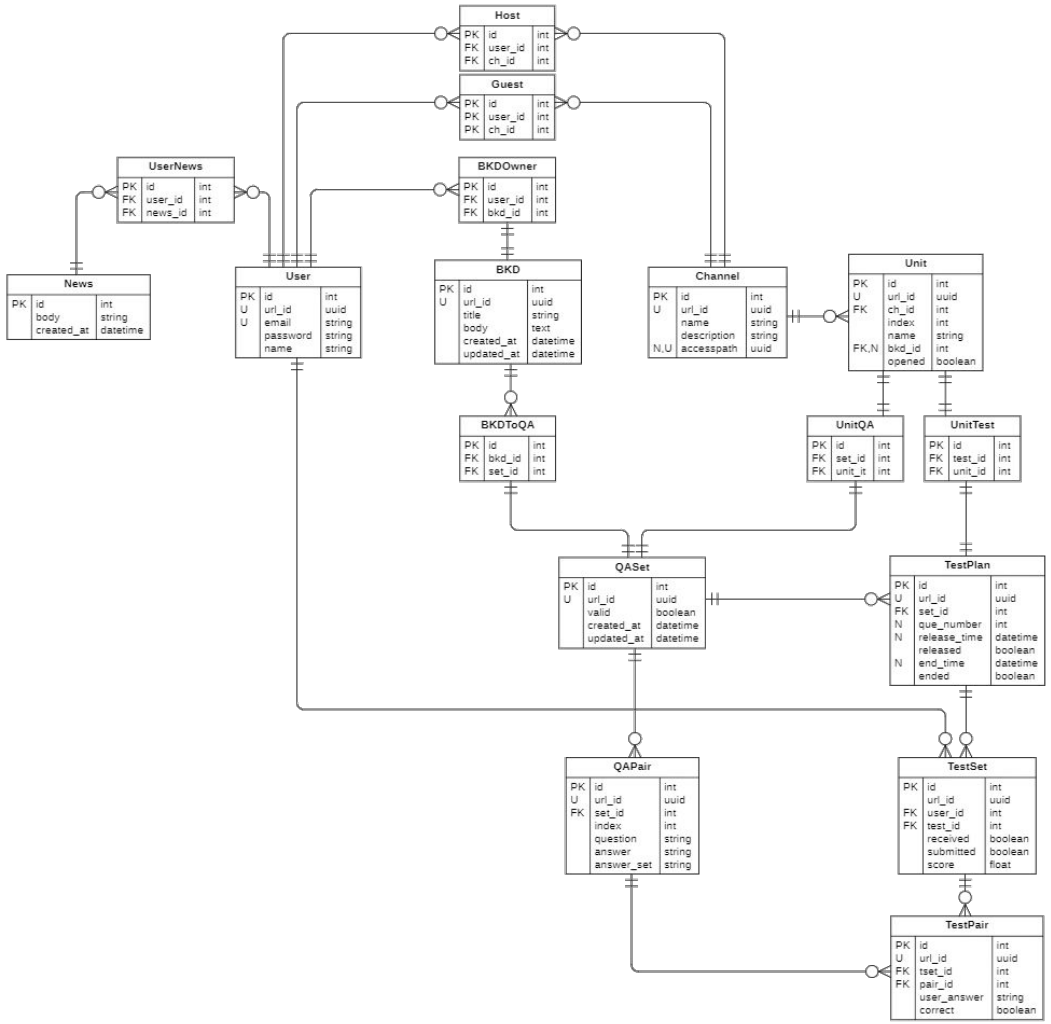
SUB SYSTEM DESCRIPTION

QG Server



SUB SYSTEM DESCRIPTION

Data Base



TRACEABILITY MATIRX

UI Frameworks

Scenario		
ID	**DESCRIPTION**	**LINK**
S.1	Make Question	WC.1,WC.11, WC.9, WC.12, WC.16, WC.17, WC.21, WC.22, WC.25
S.2	Solve Quesion	WC.25

APP SERVER		
ID	**DESCRIPTION**	**LINK**
WC.1	SignIn	DB.1.2
WC.11	Refresh	DB.1.2, DB.2.2, DB.4.2, DB.11.2, DB.12.2, DB.13.2, DB.8.2
WC.9	RequestChannel	DB.2.2
WC.12	RequestUnit	DB.3.2
WC.16	CreateBKD	DB.4.3
WC.17	EditBKD	DB.4.2, DB.4.4
WC.20	RequestQG	DB.4.2, DB.5.3, DB.5.4, DB.6.3, DB.6.4, QG.1
WC.21	VerifyQuestion	DB.6.2, DB.6.5
WC.22	MakeReservation	DB.7.3, DB.7.4, DB.8.3, DB.8.4
WC.25.	RequestPaper	DB.7.2, DB.8.2, DB.5.2, DB.6.2, DB.9.3, DB.9.4

QG	
ID	**DESCRIPTION**
QG.1	qg_per_bkd (App Server - QG Server)

DB	
ID	**DESCRIPTION**
DB.1.2	User.objects.get()
DB.2.2	Channel.objects.get()
DB.4.2	BKD.objects.get()
DB.11.2	Guest.objects.get()
DB.12.2	Host.objects.get()
DB.13.2	BKDOwner.objects.get()
DB.8.2	TestSet.objects.get()
DB.3.2	Unit.objects.get()
DB.4.3	BKD.objects.create()
DB.4.4	BKD.save90
DB.5.3	QASet.objects.create()
DB.5.4	QASet.save()
DB.6.3	QAPair.objects.create()
DB.6.4	QAPair.save()
DB.6.2	QAPair.objects.get()
DB.6.5	QAPair.delete()
DB.7.3	TestPlan.objects.create()
DB.7.4	TestPlan.save()
DB.8.3	TestSet.objects.create()
DB.8.4	TestSet.save()
DB.7.2	TestPlan.objects.get()
DB.5.2	QASet.objects.get()
DB.9.3	TestPair.objects.create()
DB.9.4	TestPair.save()