Come By (SNS Program)

Team #1

강수지(152660) 김서현(154339) 손휘진(154338) 우영주(152799)
Dept. Electronics and Computer Engineering
Chonnam National University
Gwangju

I. Motivation

관심이나 활동을 공유하는 사람들 사이의 교호적 관계망이나 교호적 관계를 구축해 주고 보여 주는 온라인 서비스 또는 플랫폼이라 정의 되어지는 SNS 2000년대에 cyworld가 네티즌의 주목을 받은 이후 꾸준히 성장해 왔다. SNS는 그 시대의 사람들의 성향에 따라 개성을 가지고 점점 바뀌어 갔는데 cyworld가 개인적인 성향이 강하였다면 이후의 SNS들인 facebook이나 instagram 등은 일방적인 교류가 아니라 소통과 관계를 중요시 하며 좀 더 간단하게 즐길 수 있는 서비스가 되었다.

우리는 이런 facebook, instagram과 같은 SNS의 기능적인 장점과 함께 이전의 cyworld와 같은 감성 또한 즐길 수 있도록 두 가지 성향을 적절히 조화시킨 SNS를 만들어서 요즘 세대뿐만 아니라 이전 세대 SNS 사용자들도 즐길 수 있는 프로그램을 완성 하고자 한다.

이 프로그램의 장점은 cyworld의 자신만의 방을 꾸며 자신을 표현하는 감성을 가지고 가되 facebook과 instagram 등의 간단하게 즐기며 쌍방향의 소통을 할 수 있는 기능을 가진다는 것이다. 다른 SNS들과 마찬가지로 언제 어디서나 사용가능하고 이전 세대의 SNS의 향수를 불러일으켜 보다 넓은 연령층을 아울러 모두가 즐길 수 있는 프로그램을 완성시키고자 한다.

II. Characteristics

일반적인 SNS 프로그램들과 마찬가지로 회원가입, 아이디와 비밀번호 찾기, 로그인과 같은 기본적인 SNS의 기능들을 포함하고 있다. 홈 화면과 뉴스피드는 나누어져 각기 다른 화면으로 존재하는 등 각각의 특징들이 타 프로그램과 비슷하게 재현된다. 홈 화면의 타임라인이나 뉴스피드에서 '좋아요' 기능과 댓글 창 기능을 가지고 있고 또한 프로필 창에서 정보를 수정하고 게시물을 포스팅 할 수 있는데 이는 instagram이나 facebook과 같은 SNS 프로그램과 거의 동일하게 제작 되었다.

특징적인 기능은 채팅 기능이라고 할 수 있다. 이 기능은 ComeBy's Room에서 실행된다. 우선 채팅방에서 보통의 1대1 채팅이 아니라 다중 채팅 방식이고 이전의 채팅 내용들은 일체 저장되지 않고 가장 최근-마지막 대화 내용만 데이터베이스에 저장된다. 채팅방의 인원은 채팅방의 주인을 포함하여 3명이고 주인이 없더라도 마지막 대화내용 기능을 통하여 자신이 남기고 싶은 말을 남길 수 있다.

III. Function Specification

ComeByProject_Client	ChatInfo	public ChatInfo (String sender, String receiver, String date, String text)	-채팅에 관한 정보를 저장하는 클래스
	Client	void connectToServer(String ip, int port)	-아이피와 포트 번호로 서버와 연결한다.(ClientThread호출)
		public static void sendToServer(String data)	-서버에게 데이터를 전달한다.
		public static void manager(String data)	-서버로부터 받아온 문자열을 분석하여 무슨 명령인지 판단하고 해당 명령에 따라 메소드나 데이터를 갱신하거나 호출한다.
		void closeAll()	-자원 정리를 위한 메소드이다.
	ClientThread	ClientThread(Socket socket, Client client)	-서버에 연결하여 데이터를 원활하게 받아오기 위한 스레드이다.
		public void run()	-스레드를 실행하는 메소드이다.
	Data	static int searchUserInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환 한다.
		static int searchPostInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환 한다.
		static int searchChatInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환 한다.
		private static String getNext(String str)	-String의 자를 다음 정보를 보낸다.
		static int tokenUser(String str, boolean mod)	-유저 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		static int tokenPost(String str, boolean mod)	-포스트 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		static int tokenChat(String str, boolean mod)	-채팅 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		static String allUserInfo(int index)	-StringBuilder에 유저의 정보를 불러와 구분기호 '|'를 넣어 저장한 후 그 값을 반환한다.
		static String allPostInfo(int index)	-StringBuilder에 포스트의 정보를 불러와 구분기호 '|'를 넣어 저장한 후 그 값을 반환한다.
		static String allChatInfo(int index)	-StringBuilder에 채팅의 정보를 불러와 구분기호 '|'를 넣어 저장한 후 그 값을 반환한다.
		static void insertUserInfo(String str)	-해당 유저 벡터의 값을 각각 데이터베이스에 저장한다.
		static void insertPostInfo(String str)	-해당 포스트 벡터의 값을 각각 데이터베이스에 저장한다.
		static void insertChatInfo(String str)	-해당 채팅 벡터의 값을 각각 데이터베이스에 저장한다.
		static boolean modifyUserInfo(String str)	-해당 유저 벡터의 값을 수정한 값을 각각 데이터베이스에 업데이트한다.
		static boolean modifyPostInfo(String str)	-해당 포스트 벡터의 값을 수정한 값을 각각 데이터베이스에 업데이트한다.
		static boolean modifyChatInfo(String str)	-해당 채팅 벡터의 값을 수정한 값을 각각 데이터베이스에 업데이트한다.

		static void deleteUserInfo(String str)	-호출 유저 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
		static void deletePostInfo(String str)	-호출 포스트 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
		static void deleteChatInfo(String str)	-호출 채팅 유저 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
	Frame		-모든 GUI창의 객체를 static으로 저장해두는 클래스
	GUI_ChangePW	public GUI_ChangePW()	-새로운 비밀번호를 설정할 수 있도록 해주는 GUI 창
	GUI_Comment	public GUI_Comment()	-게시물의 댓글을 작성하고 게시물 수정, 삭제가 가능한 창
		void addLog(String log)	-게시물의 댓글을 아래로 스크롤할수 있게 하는 메소드
		void post()	-댓글을 작성하면 벡터 위치값을 받아와서 데이터베이스와
		void refresh()	-댓글을 작성하였을 때 댓글의 내용을 댓글 박스 안에 불러오는 메소드
		void reset()	-댓글을 작성 후 댓글 텍스트 필드에 남아있는 내용을 초기화 하는 메소드
		void like()	-댓글창 '좋아요' 버튼을 누르면 '좋아요'버튼의 색이 진하게 바뀌고 서버에 게시물의 '좋아요' 한 자신의 아이디를 보낸다. -'좋아요'를 취소하고 싶을 때에는 색칠된 버튼을 다시 누르면 색이 원상태로 돌아오고 서버에게 '좋아요'정보를 삭제하라고 요청한다.
	GUI_FindIDPW	public GUI_FindIDPW()	-아이디 또는 비밀번호를 찾을 때 사용되는 GUI창
	GUI_FindPW	public GUI_FindPW()	-비밀번호를 찾을 때 사용되는 GUI창
	GUI_Home_Other	public GUI_Home_Other()	-홈화면의 GUI를 생성하는 메소드 -프로필 설정하는 메소드로 이동할 수 있다. -로그아웃을 할 수 있는 버튼이 존재한다. -뉴스피드로 이동할 수 있는 버튼이 존재한다. -팔로우 하는 사람의 수와 팔로우 하는 사람의 정보를 리스트로 보여주는 프레임으로 이동하는 버튼이 존재한다. -프로필 사진과 한 줄 소개를 볼 수 있다. -타임라인에 자신의 게시물을 등록하는 버튼이 존재한다. -타임라인에 자신의 등록한 최근 게시물 순서대로 볼 수 있고 앞/뒤 버튼을 통해서 게시물 이동을 해서 이전에 등록한 게시물을 볼 수 있다.
		void search(String input)	-다른 사람의 계정으로 이동 할 수 있는 검색 창 -리스트로 구성 되어 이동하고 싶은 계정을 클릭하면 그 계정으로 이동하는 메소드를 호출하고 그 계정의 정보를 데이터 베이스에서 불러온다.
		void refresh()	-회원가입 했을 때 저장 했던 홈의 계정 정보를 데이터 베이스에서 불러오는 메소드
		void follow()	-다른 사람의 계정에서 사용자가 팔로우 버튼을 누르면 팔로우 버튼 색이 진해지고 팔로우 한 사람의 정보를 서버에 보내 데이터베이스에 저장되도록 요청한다.

			-팔로우를 하고 싶지 않을 때 칠해진 팔로우 버튼을 누르면 버튼 색이 원 상태로 바뀌고 서버에게 데이터베이스의 팔로우 정보를 삭제하도록 요청한다.
	GUI_Home	public GUI_Home()	-홈화면의 GUI를 생성하는 메소드 -프로필 설정하는 메소드로 이동할 수 있다. -로그아웃을 할 수 있는 버튼이 존재한다. -뉴스피드로 이동할 수 있는 버튼이 존재한다. -팔로우 하는 사람의 수와 팔로우 하는 사람의 정보를 리스트로 보여주는 프레임으로 이동하는 버튼이 존재한다. -프로필 사진과 한 줄 소개를 볼 수 있다. -타임라인에 자신의 게시물을 등록하는 버튼이 존재한다. -타임라인에 자신의 등록한 최근 게시물 순서대로 볼 수 있고 앞/뒤 버튼을 통해서 게시물 이동을 해서 이전에 등록한 게시물을 볼 수 있다.
		void search(String input)	-다른 사람의 계정으로 이동 할 수 있는 검색 창 -리스트로 구성 되어 이동하고 싶은 계정을 클릭하면 그 계정으로 이동하는 메소드를 호출하고 그 계정의 정보를 데이터 베이스에서 불러온다.
		void refresh()	-회원가입 했을 때 저장 했던 사용자 정보를 데이터 베이스에서 불러오는 메소드
		void like()	-타임라인에 '좋아요' 버튼을 누르면 '좋아요'버튼의 색이 진하게 바뀌고 서버에 게시물의 '좋아요' 한 자신의 아이디를 보낸다. -'좋아요'를 취소하고 싶을 때에는 색칠된 버튼을 다시 누르면 색이 원상태로 돌아오고 서버에게 '좋아요'정보를 삭제하라고 요청한다.
		void follow()	-팔로우 리스트로 이동하는 버튼 -팔로우 리스트 프레임을 불러온다.
	GUI_Join	public GUI_Join()	-회원가입에 사용되는 GUI창
		public void join()	-유저 정보를 입력받고 회원가입을 하게 해주는 메소드
		void reset()	-모든 입력란의 내용을 지워주는 메소드
	GUI_Login	public void init()	-초기 프레임을 띄우는 메소드
		public GUI_Login()	-초기화면을 나타내는 GUI창
		private void login()	-유저 정보를 불러와서 이에 해당하는 계정이 있으면 로그인을 하는 메소드
	GUI_NewsFeed	public GUI_NewsFeed()	-뉴스피드 창 -프로필 설정하는 메소드로 이동할 수 있다. -로그아웃을 할 수 있는 버튼이 존재한다. -홈화면로 이동할 수 있는 버튼이 존재한다. -팔로우 하는 사람의 수와 팔로우 하는 사람의 정보를 리스트로 보여주는 프레임으로 이동하는 버튼이 존재한다. -프로필 사진과 한 줄 소개를 볼 수 있다. -타임라인에 자신의 게시물을 등록하는 버튼이 존재한다. -타임라인에 자신의 등록한 최근 게시물 순서대로 볼 수 있고 앞/뒤 버튼을 통해서 게시물 이동을 해서 이전에 등록

			한 게시물을 볼 수 있다.
		void refresh()	-사용자가 게시글을 포스팅 했을 경우 포스팅한 게시물 내용과 사용자 정보를 데이터 베이스에서 불러오는 메소드
		void like()	-타임라인에 '좋아요' 버튼을 누르면 '좋아요'버튼의 색이 진하게 바뀌고 서버에 게시물의 '좋아요' 한 자신의 아이디를 보낸다. -'좋아요'를 취소하고 싶을 때에는 색칠된 버튼을 다시 누르면 색이 원상태로 돌아오고 서버에게 '좋아요'정보를 삭제하라고 요청한다.
		void follow()	-팔로우 리스트로 이동하는 버튼 -팔로우 리스트 프레임을 불러온다.
		void search(String input)	-다른 사람의 계정으로 이동 할 수 있는 검색 창 -리스트로 구성 되어 이동하고 싶은 계정을 클릭하면 그 계정으로 이동하는 메소드를 호출하고 그 계정의 정보를 데이터 베이스에서 불러온다.
	GUI_Post	public GUI_Post()	-게시물에 관한 GUI 정보를 생성하는 메소드
		void post()	-게시물 등록하기 버튼을 누르면 실행되는 메소드 -게시글을 서버에 보낸 후 서버에서 홈화면에 게시글을 올려준다.
		void reset()	-게시물 등록한 정보를 갱신해주는 메소드
	GUI_Profile	public GUI_Profile()	-프로필에 관한 GUI 정보를 생성하는 메소드
		public void change()	-프로필 정보를 수정하는 버튼을 누를 시에 실행되는 메소드 -프로필을 수정하는 정보와 한 줄 소개, 캐릭터 설정에 대한 정보를 서버에 보내고 데이터베이스에 정보를 갱신하라고 요청한다.
		void refresh()	-프로필을 수정하기 전에 저장된 정보를 디비에서 불러오는 메소드
	PostInfo	public PostInfo(String date, String id, String text, String loc, String c1, String c2, String liker)	-게시물 관리 시스템에 필요한 필드들을 가지는 클래스
	UserInfo	public UserInfo(String id, String email, String name, int pwQuestion, String introduce, int animal, int followerNum, int postNum, String followee, int port)	-유저 정보 관리 시스템에 필요한 필드들을 가지는 클래스
Come ByProject_Server	ChatInfo	public ChatInfo(String sender, String receiver, String date, String text)	-채팅 관리 시스템에 필요한 필드들을 가지는 클래스
	DB	void connectDB()	-데이터베이스와 연결하는 역할을 한다.
		public void inputQuery(String s)	-데이터베이스에 데이터 입력한다.
		void getDBInfo()	-데이터베이스의 값을 벡터에 저장한다.(이미 제작되어있는 UserInfo, PostInfo, ChatInfo처럼 데이터를 저장할 클래스를 만들고 벡터화한다.)
		int searchUserInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.

		int searchPostInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		int searchChatInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		int searchEmailInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		int searchNameInfo(int i, String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		int searchPostCommentInfo(String str)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		String getUserIDInfo(int index)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		int getUserPWFindQInfo(int index)	-id/code 맞는 id/code를 찾아서 벡터의 번호 반환한다.
		private String getNext(String str)	-String의 자를 다음 정보를 보낸다.
		int tokenUser(String str, boolean mod)	-유저 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		int tokenPost(String str, boolean mod)	-포스트 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		int tokenChat(String str, boolean mod)	-채팅 코드에 저장된 string 값을 각각의 정보를 받아오기 위해 자른다.
		String getUserInfo(int index)	-StringBuilder에 유저의 정보를 불러와 구분기호 'I'를 넣어 저장한 후 그 값을 반환한다.
		String getPostInfo(int index)	-StringBuilder에 포스트의 정보를 불러와 구분기호 'I'를 넣어 저장한 후 그 값을 반환한다. (
		String getChatInfo(int index)	-StringBuilder에 채팅의 정보를 불러와 구분기호 'I'를 넣어 저장한 후 그 값을 반환한다.
		String insertUserInfo(String str)	-해당 유저 벡터의 값을 각각 데이터베이스에 저장한다.
		void insertPostInfo(String str)	-해당 포스트 벡터의 값을 각각 데이터베이스에 저장한다.
		void insertChatInfo(String str)	-해당 채팅 벡터의 값을 각각 데이터베이스에 저장한다.
		boolean isPWCorrect(String str, String strPw, boolean find)	-비밀번호확인 질문의 답과 일치하는지 확인한다.
		boolean modifyUserInfo(String str)	-수정할때 현재비밀번호확인을 확인하기 위해 isPWCorrect를 호출하고 일치하면 true, 일치하지 않으면 false를 호출한다.
		void modifyPostInfo(String str)	-해당 포스트 벡터의 값을 수정한 값을 각각 데이터베이스에 업데이트한다.
		void modifyChatInfo(String str)	-해당 채팅 벡터의 값을 수정한 값을 각각 데이터베이스에 업데이트한다.
		void deleteUserInfo(String str)	-searchUserInfo(str) 호출 유저 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
		void deletePostInfo(String str)	-searchUserInfo(str) 호출 포스트 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
		void deleteChatInfo(String str)	-searchUserInfo(str) 호출 채팅 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.
		void deletePostCommentInfo(String str)	-searchUserInfo(str) 호출 게시물의 댓글 벡터 위치값 받아와서 데이터베이스와 벡터에서 삭제한다.

		int userInfoNum()	-등록한 유저의 수를 반환한다.
		int postInfoNum()	-등록한 포스트의 수를 반환한다.
		int chatInfoNum()	-등록한 채팅의 수를 반환한다.
	GUI_ServerManager	public void init()	-서버매니저 창을 실행하기 위한 메소드이다.
		public GUI_ServerManager()	-서버매니저 GUI에 스윙 컴포넌트들을 포함시키는 생성자이다.
	PostInfo	public PostInfo(String date, String id, String text, String loc, String c1, String c2, String liker)	-게시물 관리 시스템에 필요한 필드들을 가지는 클래스
	UserInfo	public UserInfo(String id, String pw, String email, String name, int pwQuestion, String pwAnswer, String introduce, int animal, int followerNum, int postNum, String followee, int port)	-회원 관리 시스템에 필요한 필드들을 가지는 클래스
	Server	public void init()	-서버 소켓을 열어 클라이언트가 접속하면 스레드 값을 저장하여 멀티 채팅이 되도록 한다.
		public Server()	-서버 소켓과 데이터베이스를 임시로 담아놓기 위한 벡터들을 초기화하고 클라이언트들이 접속하면 사용할 스레드의 정보를 담아놓기 위해 벡터를 데이터베이스의 사람 수만큼 생성한다.
		public synchronized void addUser()	-새로운 계정 정보가 추가되면(회원가입) 담을 벡터의 자리를 늘린다.
		public synchronized void addVec(Thread userThread)	-로그인 이전의 스레드를 임시로 담아놓는 벡터에 스레드를 저장한다.
		public synchronized void addVec(Vector vec,int index, Object obj)	-로그인을 하면 임시로 저장된 스레드 값을 정식으로 벡터에 옮겨준다.
		public synchronized void selectedCast(String message, String clients)	-clients로 선택된 사용자 아이디를 받아와 선택된 사용자에게만 데이터를 전송한다.
		public synchronized void manager(String data)	-클라이언트로부터 받아온 데이터를 구분 기호에 따라 토크나이저로 구분하여 해당 요청에 따라 데이터베이스의 값을 갱신하거나 오류를 보내거나 정보를 보내준다.
		public synchronized void broadCast(String message)	-스레드로 접속된 모든 사용자에게 데이터를 전달한다.
		public synchronized String getElement(Vector getVec, Vector vec, String str)	-벡터에 있는 값들을 다른 벡터의 한 위치에 구분 기호를 포함하여 옮겨준다.
		public synchronized void deleteFromRoom(int index)	-사용자가 다른 위치로 이동하는 경우 위치 정보와 사용자가 있던 방의 정보, 이동할 방의 정보를 갱신한다.
	ServerThread	public ServerThread(Socket socket,Server server)	-소켓과 ReaderWriter를 초기화하는 생성자이다.
		public void run()	-클라이언트의 연결을 받아 데이터를 원활하게 할 수 있도록 스레드를 실행한다.
		public void	-현재 연결 중인 클라이언트에게 데이터를 보내준다.

		sendMessage(String server_message)	
		public void closeServer()	-자원 정리를 위한 메소드이다.

IV. Requirement Lists

1. 회원가입 시 아이디 중복여부, 아이디와 비밀번호의 길이제한, 비밀번호와 비밀번호 확인의 일치 여부, 이메일 형식을 확인받을 수 있다.
2. 회원가입 시 본인 확인 문항을 설정할 수 있다.
3. 이름과 이메일을 입력하여 아이디를 찾을 수 있다.
4. 비밀번호 찾기 질문에 대한 답을 입력하여 비밀번호를 찾을 수 있다.
5. 로그인 실패 시 오류 사항을 알림 받을 수 있다.
6. 회원은 홈 화면에서 ComeBy's room을 통해 다른 회원들과 그룹 채팅을 할 수 있다.
7. 회원은 채팅의 마지막 내용을 이후에 볼 수 있다.
8. 회원은 프로필 화면에서 자신의 프로필을 설정할 수 있고 한 줄 소개를 할 수 있다.
9. 회원은 홈 화면에서 자신의 타임라인에 게시물을 추가할 수 있다.
10. 회원은 프로필 화면에서 자신을 팔로우 하는 사람의 리스트 목록을 볼 수 있다.
11. 회원은 홈 화면에서 다른 회원을 검색하여 다른 회원의 홈 화면에 들어가고 팔로우할 수 있다.
12. 회원은 홈 화면에서 뉴스피드로 이동할 수 있고 뉴스피드에서 홈 화면으로 이동할 수 있다.
13. 회원은 게시물에 '좋아요'를 표시할 수 있고 댓글을 작성할 수 있다.
14. 회원은 뉴스피드에서 팔로우 한 사람의 게시물을 볼 수 있다.
15. 회원은 새로 고침 버튼을 통해 게시물 등의 정보를 갱신할 수 있다.
16. 자신이 작성한 게시물은 자신의 타임라인에 순서대로 볼 수 있다.
17. 회원들의 비밀번호는 암호화 되어 있어야 하며 관리자가 비밀번호를 직접적으로 확인할 수 없다.
18. 관리자는 데이터베이스에 저장된 정보를 확인할 수 있다.

V. Usecases

I. 초기 화면

1. 회원가입
   - 아이디, 비밀번호, 비밀번호확인, 이메일, 이름, 본인확인질문, 본인확인답안 항목을 입력받고 회원가입 버튼을 통해 계정을 생성한다.
   - 아이디와 비밀번호, 이메일은 주어진 형식에 맞춰 입력하지 않으면 회원가입 버튼을 누를 시에 팝업 메시지로 에러가 표시된다.
   - 본인확인질문은 예시가 주어져 고를 수 있도록 한다.
   - 모든 항목은 길이 제한을 두어 이를 초과하면 더 이상 입력되지 않도록 한다.
2. 아이디 찾기
   - 이메일, 이름을 입력받고 확인 버튼을 통해 해당 이메일과 이름의 아이디를 팝업 메시지로 출력한다.
   - 해당 이메일과 이름이 서로 맞지 않거나 없는 경우 팝업 메시지로 에러가 표시된다.
3. 비밀번호 찾기
   - 아이디, 이름, 이메일을 입력받고 확인 버튼을 통해 해당 아이디, 이메일의 본인확인질문을 팝업 메시지로 출력하고 질문에 해당하는 답을 입력받을 수 있도록 한다.
   - 아이디, 이름, 이메일이 서로 맞지 않거나 없는 경우 팝업 메시지로 에러가 표시된다.
   - 입력받은 본인확인답안이 정답이 아닐 경우 팝업 메시지로 에러가 표시된다.
   - 입력받은 본인확인답안이 정답일 경우 비밀번호를 재설정하도록 한다.
   - 재설정한 비밀번호가 형식에 어긋날 경우 팝업 메시지로 에러가 표시된다.
4. 로그인
   - 아이디와 비밀번호를 입력받고 로그인 버튼을 누르면 해당 아이디와 비밀번호가 일치하는 경우 계정에 접속된다.
   - 아이디와 비밀번호가 서로 맞지 않거나 없는 아이디일 경우 팝업 메시지로 에러가 표시된다.

II. 홈 화면

1. 프로필

① 설정 버튼(새 창)

- 계정의 정보 중 이름, 프로필 사진, 한줄 소개, 비밀번호, 본인확인질문, 본인확인답안, 캐릭터 항목의 변경 사항을 새로 입력하여 변경 버튼을 통해 정보를 변경할 수 있다.
- 비밀번호는 주어진 형식에 맞춰 입력하지 않으면 변경 버튼을 누를 시에 팝업 메시지로 에러가 표시된다.
- 모든 항목은 길이 제한을 두어 이를 초과하면 더 이상 입력되지 않도록 한다.
- 캐릭터는 4개의 선택지 중 하나를 고를 수 있다.
- 프로필 사진은 파일 탐색기를 통해 내 컴퓨터의 사진을 가져와 등록할 수 있도록 한다.
- 사용자의 계정의 홈 화면일 경우만 표시된다.

② 계정 검색

- 프로그램을 사용하는 지인의 이름 혹은 아이디를 입력하고 검색 버튼을 통해 해당 이름 혹은 아이디의 계정을 검색하여 리스트로 검색된 계정을 보여준다.
- 리스트로 표시된 계정을 클릭하여 해당 계정의 홈 화면으로 이동할 수 있다.
- 해당 이름 혹은 아이디의 계정이 없는 경우 리스트에 아무 출력도 나오지 않는다.

③ 로그아웃 버튼

- 로그아웃 버튼을 통해 계정의 접속이 종료된다.

④ 팔로우/언팔로우 버튼

- 다른 계정 사용자의 홈 화면일 경우 팔로워 리스트 버튼 대신에 해당 계정을 팔로우/언팔로우 할 수 있는 버튼이 존재한다.
- 해당 계정을 팔로우 중인 경우 언팔로우 버튼, 언팔로우 중인 경우 팔로우 버튼이 표시된다.
- 팔로우 버튼을 누르면 해당 계정이 포스팅한 게시물을 뉴스피드에서 볼 수 있고 언팔로우 버튼을 누르면 해당 계정이 포스팅한 게시물이 뉴스피드에서 사라지게 된다.

⑤ 뉴스피드 화면 버튼

- 뉴스피드 화면 버튼을 통해 뉴스피드로 이동한다.

2. 타임라인

① 게시물확인(새 창)

- 뉴스피드 화면의 게시물 확인과 동일하다.

② 포스팅 버튼

- 글을 작성하여 등록 버튼을 통해 사용자 계정의 타임라인과 사용자를 팔로우하는 계정들의 뉴스피드에 게시된다.
- 글만 입력하여 게시할 수 있다.
- 글은 길이 제한을 두어 이를 초과하면 더 이상 입력되지 않도록 한다.

③ 다음/이전 게시물이동 버튼

- 게시물은 사용자와 사용자가 팔로우하는 다른 계정의 게시물을 작성된 시간 순서대로 정렬하여 표시한다.
- 한 번에 세 개의 게시물을 볼 수 있으며 다음/이전 게시물이동 버튼을 통해 다음 시간 순서/이전 시간 순서의 게시물을 볼 수 있다.

3. 채팅

① ComeBy's room

- 사용자의 캐릭터가 ComeBy's room에 위치하여 사용자가 전송한 메시지를 말풍선으로 표시한다.
- 다른 계정의 사용자가 사용자의 홈 화면에 접속한 경우 다른 계정 사용자의 캐릭터 또한 ComeBy's room에 위치하여 전송한/된 메시지를 표시한다.
- 한 번에 채팅을 할 수 있는(ComeBy's room에 캐릭터가 위치할 수 있는) 수는 제한이 되어 초과되면 채팅은 할 수 없다.

② 채팅전송

- 보낼 메시지를 입력받고 전송 버튼을 통해 ComeBy's room에 메시지를 출력한다. 이때 사용자의 홈 화면에 접속 중인 다른 계정의 사용자 또한 메시지를 확인할 수 있다.
- 새로운 메시지를 보내면 이전의 메시지는 삭제된다.
- 사용자의 마지막으로 보내진 메시지는 새로운 메시지가 ComeBy's room에 입력되기 전까지 남아있다.

③ 마지막채팅확인

- 다른 계정의 사용자가 사용자에게 전송한 마지막 메시지는 다른 계정의 사용자가 사용자의 홈 화면을 나가더라도 리스트에 남아 기록된다. 동일한 다른 계정의 사용자가 다시 사용자에게 메시지를 전송하면 이전의 마지막 메시지는 사라지고 새로운 마지막 메시지가 갱신된다.

III. 뉴스피드 화면

1. 게시물확인(새 창)

① 댓글

- 댓글을 입력하고 입력 버튼을 통해 댓글을 달 수 있다.
- 댓글은 길이 제한을 두어 이를 초과하면 더 이상 입력되지 않도록 한다.

② 게시물수정 버튼

- 해당 게시물을 작성한 사용자만이 게시물을 수정할 수 있다.
- 게시물수정 버튼을 통해 홈 화면의 타임라인에서의 포스팅 버튼과 같은 기능을 할 수 있다.

③ 게시물삭제 버튼

- 해당 게시물을 작성한 사용자만이 게시물을 삭제할 수 있다.
- 게시물삭제 버튼을 통해 게시물을 삭제한다.

④ 좋아요 버튼

- 게시물에 공감을 나타내기 위한 버튼으로 버튼을 누르면 게시물의 좋아요 수가 1증가한다.
- 이미 좋아요 버튼을 누른 게시물을 다시 좋아요 버튼을 누르면 좋아요 를 취소할 수 있다.
- 좋아요 버튼을 누르면 좋아요 표시가 변경되어 좋아요 버튼을 눌렀다는 것을 알 수 있다.

2. 좋아요 버튼

- 게시물확인의 좋아요 버튼과 기능이 동일하다.

3. 다음/이전게시물이동 버튼

- 홈 화면의 다음/이전게시물이동 버튼과 기능이 동일하다.

4. 프로필

- 홈 화면의 프로필과 동일하다.

5. 홈 화면 버튼

- 홈 화면 버튼을 통해 홈 화면으로 이동한다.

IV. 서버(관리자)

1. 전체 계정 정보 리스트

- 아이디, 비밀번호, 비밀번호확인, 이메일, 이름, 본인확인질문(번호), 본인확인답안, 한줄 소개, 캐릭터(번호), 팔로우 정보(팔로우한 계정 아이디, 총 팔로워 수), 총 게시물 수, 접속 여부, 마지막 채팅 내용 항목 (보낸 계정 아이디, Date/Time, TEXT)을 리스트로 표시한다.
- 새로운 정보가 갱신될 때마다 함께 갱신되도록 하여 변경 사항을 볼 수 있도록 한다.

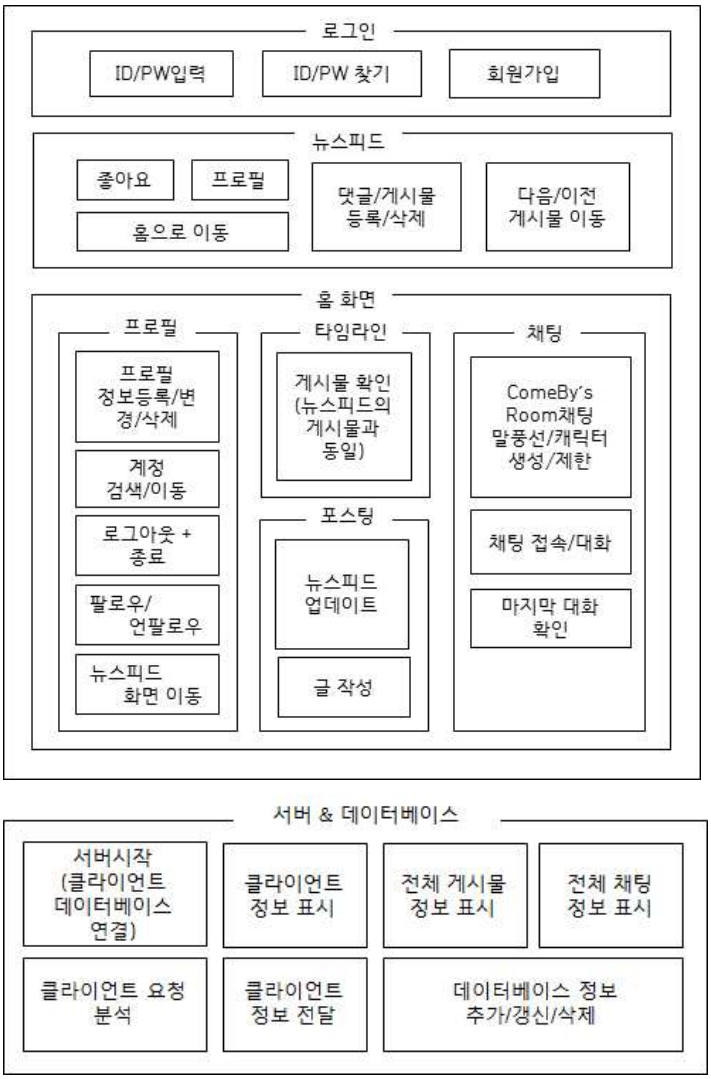
2. 전체 게시물 정보 리스트

- Date/Time, 아이디, TEXT, 게시물코드, 댓글코드를 리스트로 표시한다.
- 새로운 정보가 갱신될 때마다 함께 갱신되도록 하여 변경 사항을 볼 수 있도록 한다.

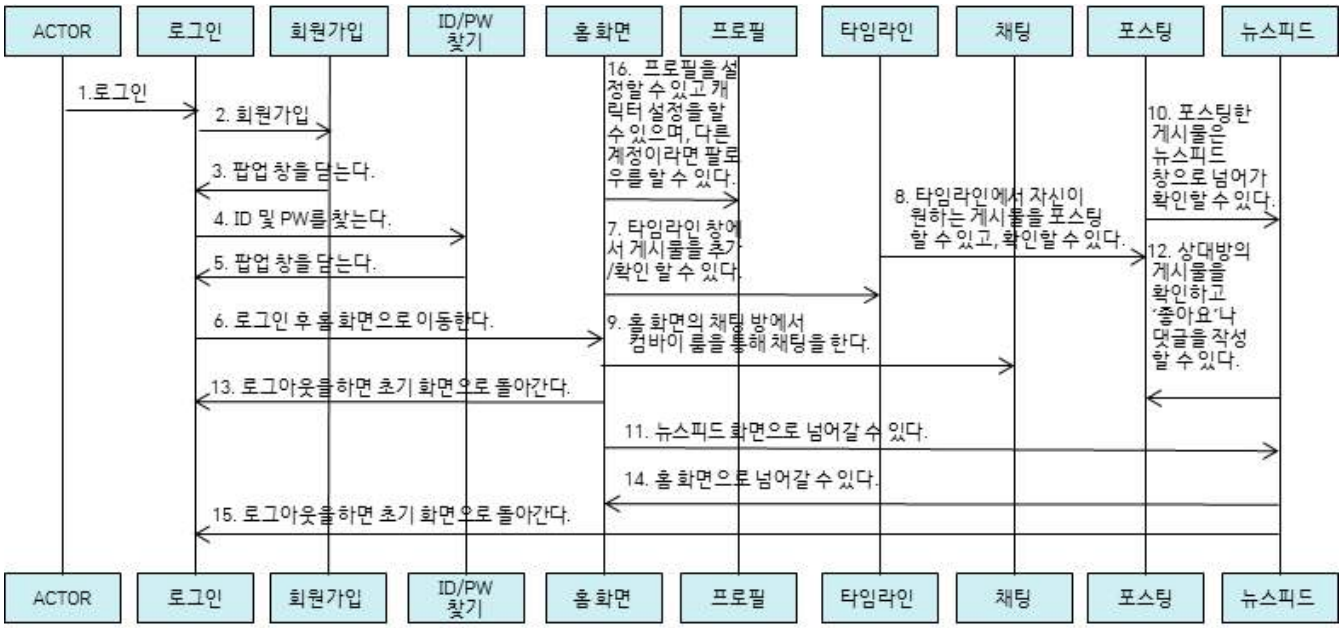
3. 항목 검색 버튼

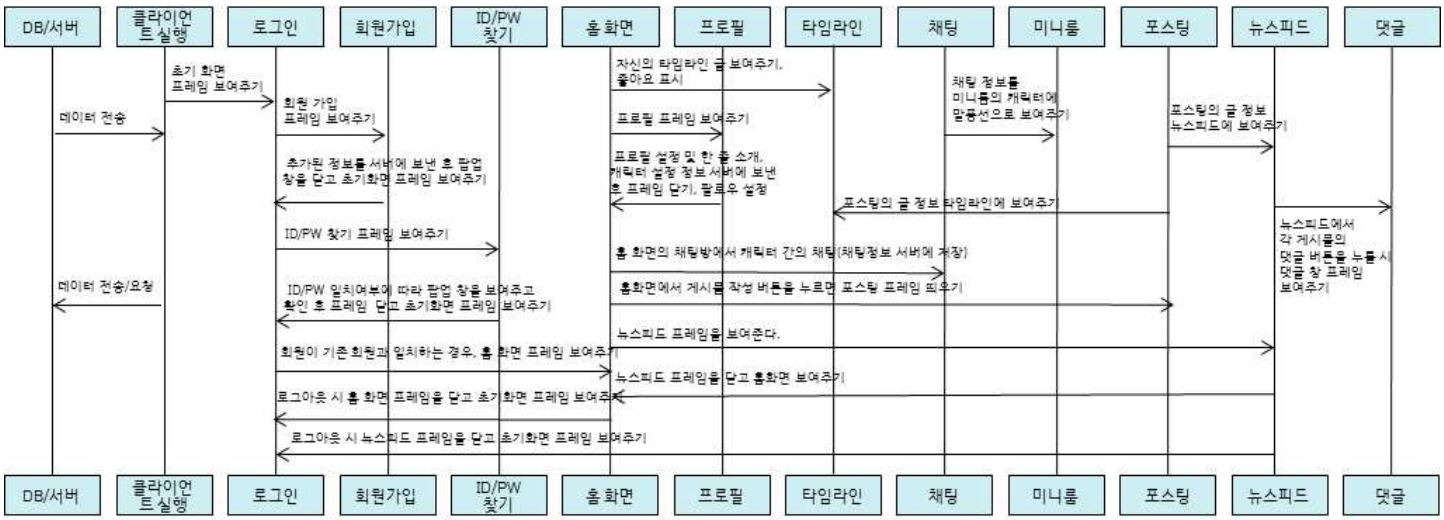
- 아이디, 이메일, 게시물 코드, 댓글 코드를 검색할 수 있도록 한다.
- 검색된 결과는 리스트로 보여준다.
- 해당 항목이 없는 경우 리스트에 아무 출력도 나오지 않는다.

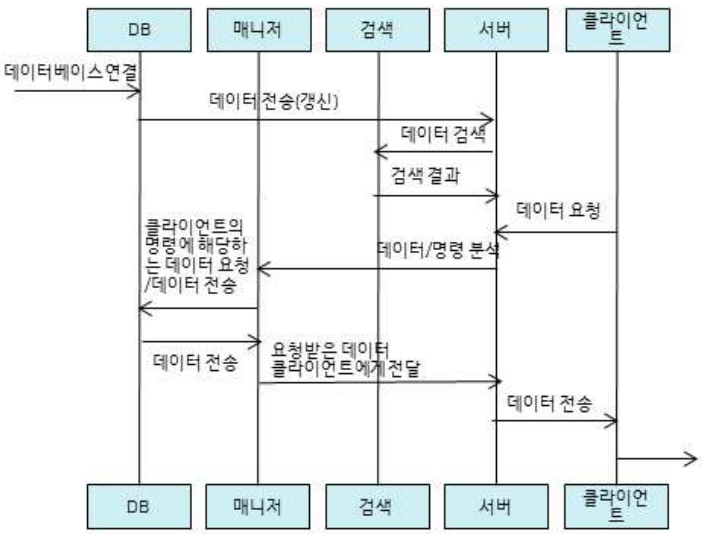
VI. Usecase Diagram



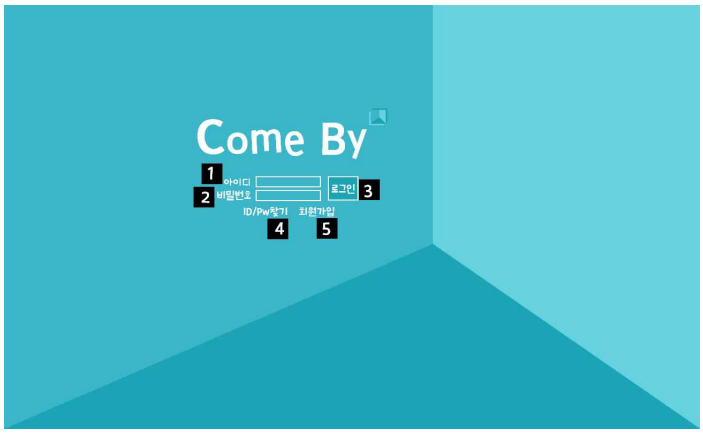
VII. Sequence Diagram



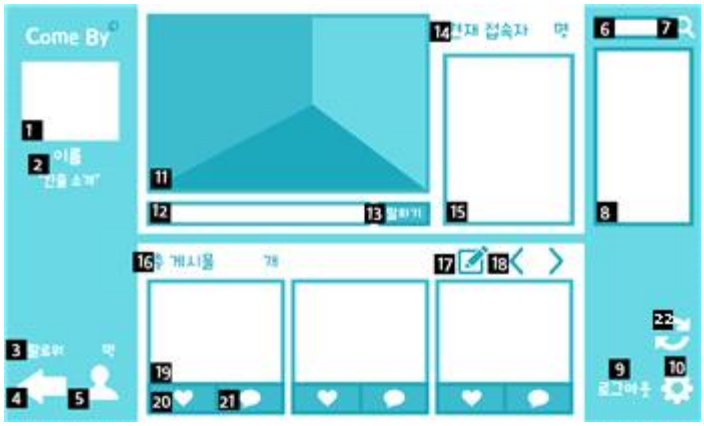




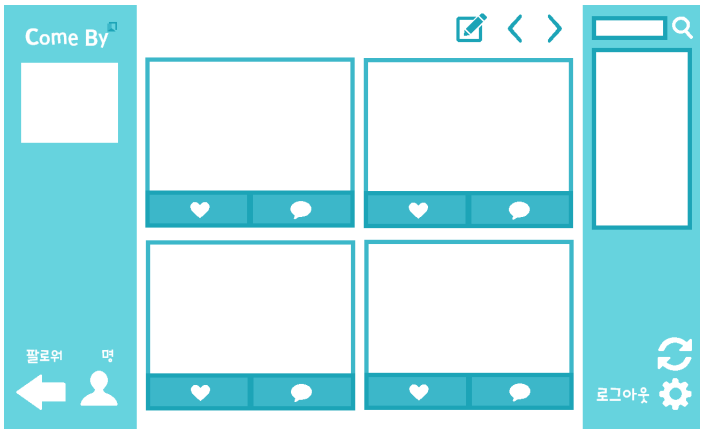
초기화면



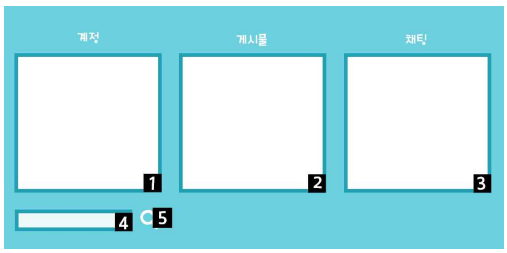
홈 화면



뉴스피드 화면



서버관리자 화면

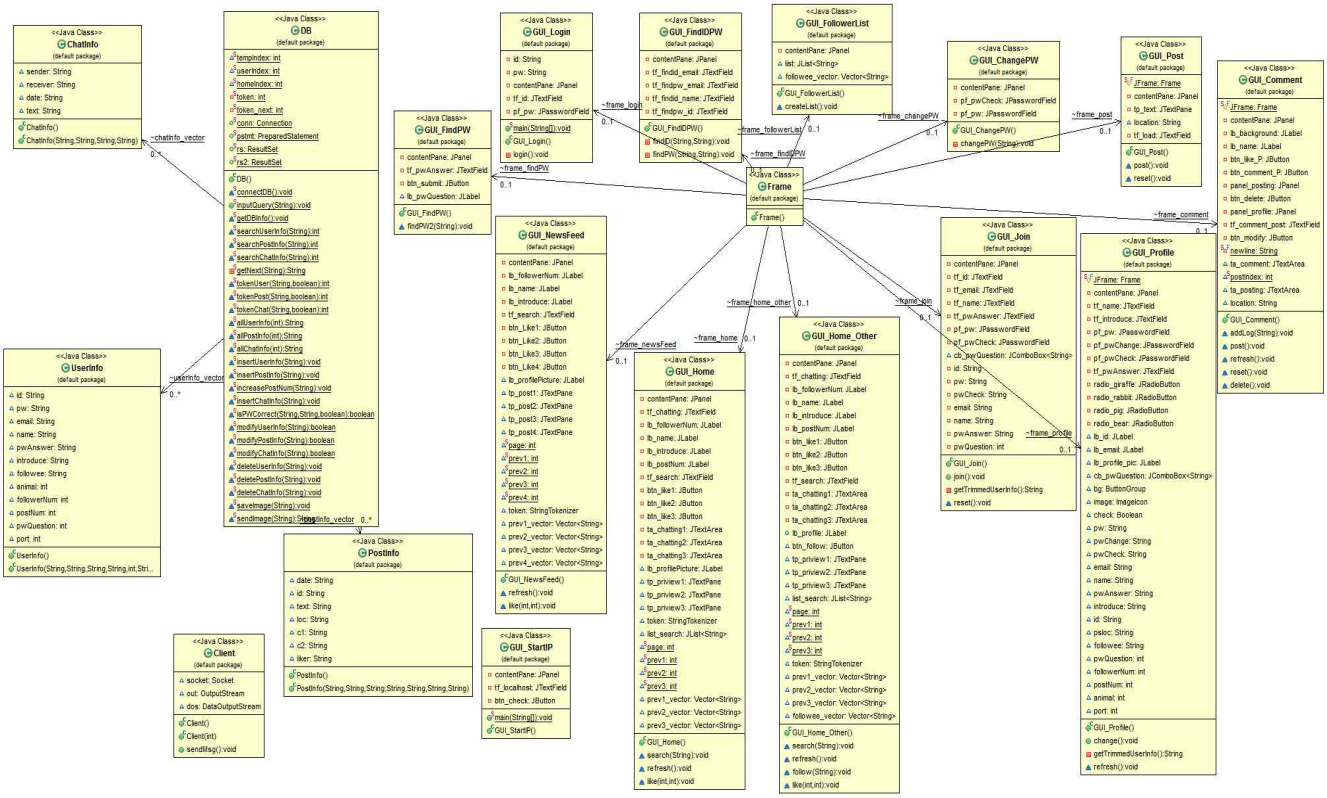


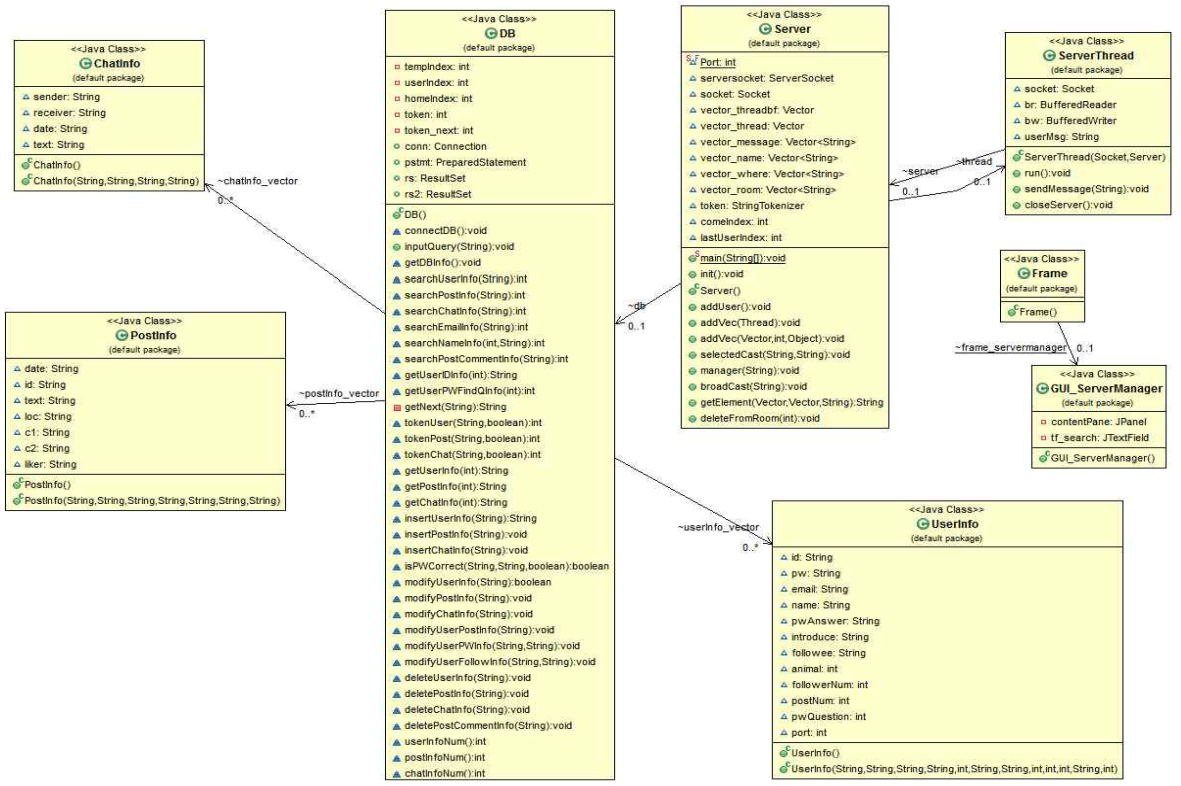
<초기화면>
[1] 아이디를 입력받는 텍스트 필드
[2] 비밀번호를 입력받는 패스워드 텍스트 필드
[3] 입력받은 [1], [2]항목으로 로그인을 하는 버튼
[4] 새 팝업 창을 통해 아이디/비밀번호를 찾을 수 있도록 하는 버튼
[5] 새 팝업 창을 통해 회원가입을 할 수 있도록 하는 버튼

<홈 화면>
[1] 프로필 사진이 표시되는 라벨
[2] 이름, 한줄 소개가 표시되는 라벨
[3] 팔로워 수를 표시해 주는 라벨
(다른 사용자의 홈인 경우 보이지 않음)
[4] 뉴스피드로 창을 이동하는 버튼
[5] 새 팝업 창으로 자신을 팔로우 한 사람의 리스트를 보여주는 버튼
(다른 사용자의 홈인 경우 팔로우 버튼)
[6] 검색 항목을 입력받는 텍스트 필드
[7] 입력받은 [6]항목으로 검색을 하는 버튼
[8] 검색한 정보를 보여주는 리스트
[9] 로그아웃하고 초기 화면으로 창을 이동하는 버튼
[10] 새 팝업 창을 통해 회원 정보 수정을 할 수 있도록 하는 버튼
[11] 캐릭터들이 채팅 말풍선과 함께 표시되는 창
[12] 보낼 메시지를 입력받는 텍스트 필드
[13] 입력받은 [12]항목을 보내는 버튼
[14] 채팅에 접속 중인 사용자 수를 보여주는 라벨
[15] 마지막 대화 내용을 시간 순으로 보여주는 리스트
[16] 총 게시물 수를 보여주는 라벨
[17] 새 팝업 창을 통해 게시물을 작성할 수 있도록 하는 버튼(다른 사용자의 홈인 경우 보이지 않음)
[18] 이전/다음 게시물을 볼 수 있도록 하는 버튼
[19] 게시물의 내용을 일부 표시하는 라벨
[20] 게시물에 '좋아요'를 1증가시키는 버튼
[21] 새 팝업 창을 통해 게시물 내용을 자세히 볼 수 있도록 하고 댓글을 남길 수 있도록 하는 버튼
[22] 창을 새로운 정보를 바탕으로 갱신하는 버튼
뉴스피드의 경우 대부분의 사항이 중복되어 생략

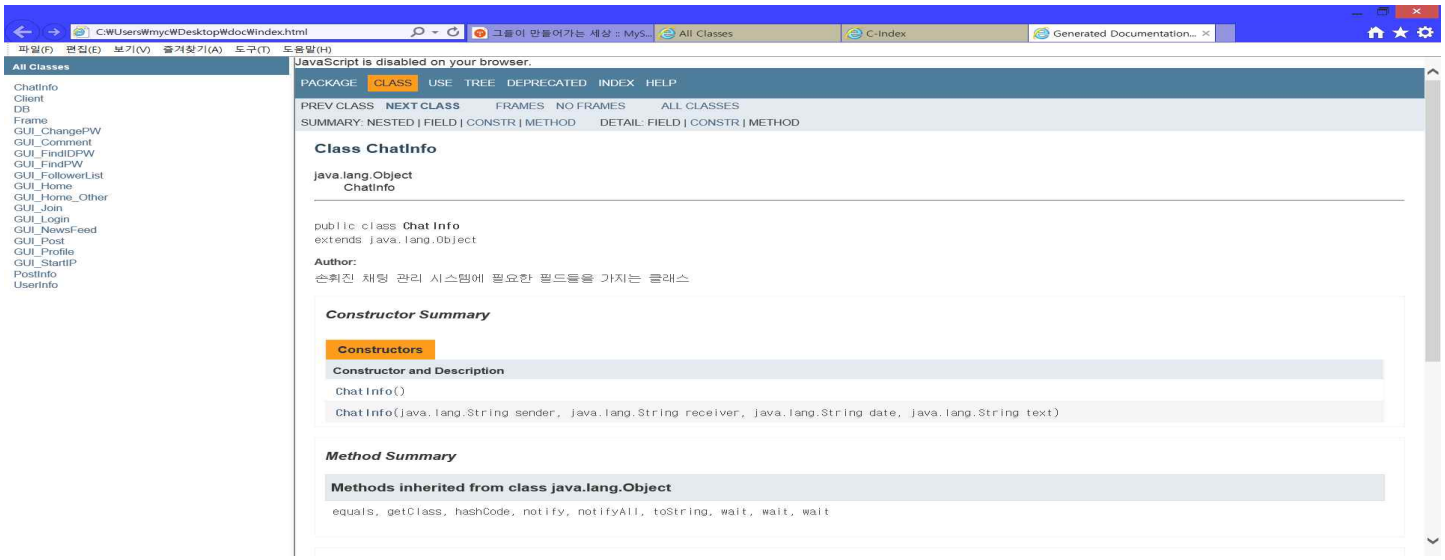
<서버관리자 화면>
[1] user DB table의 내용을 표시하는 리스트
[2] posting DB table의 내용을 표시하는 리스트
[3] chatting DB table의 내용을 표시하는 리스트
[4] 검색 항목을 입력받는 텍스트 필드
[5] 입력받은 [4]항목으로 검색을 하는 버튼

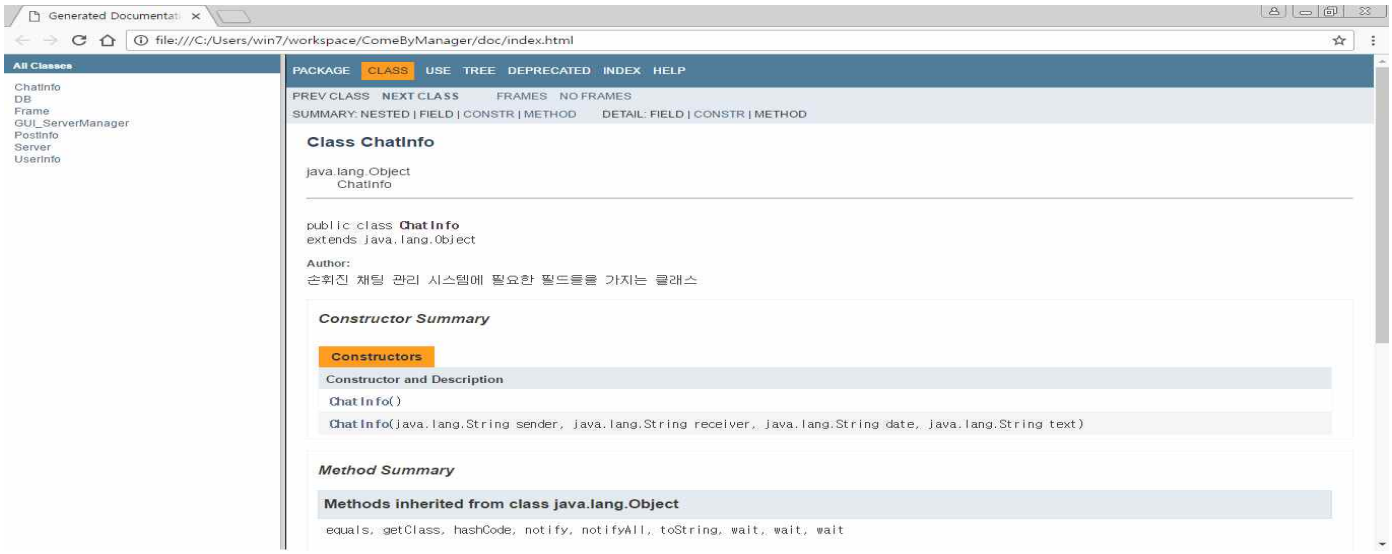
IX. Class Diagram

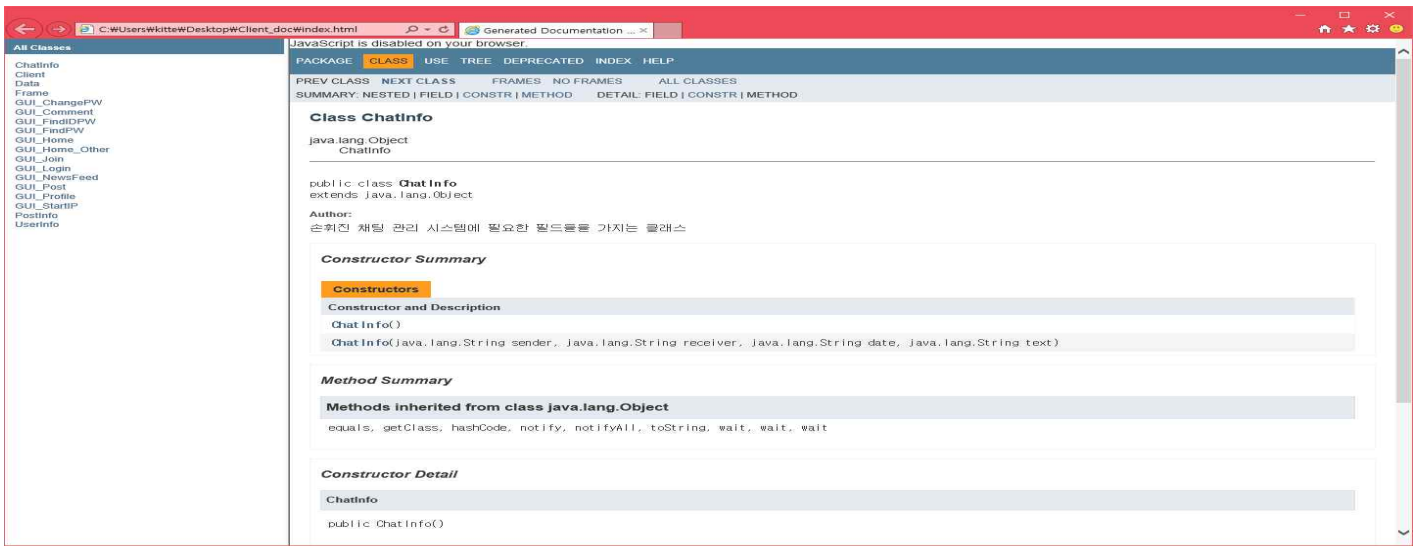




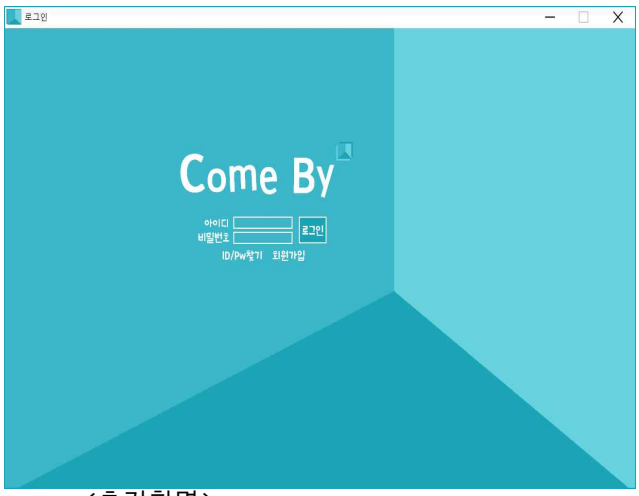
X. Screenshot of API Manual







XI. Screen shot of Demo Version

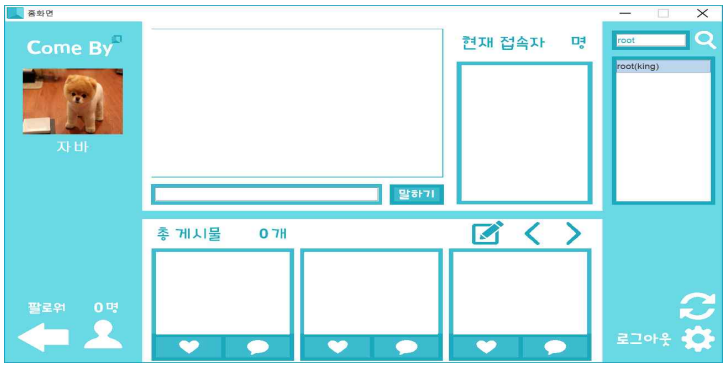


<초기화면>

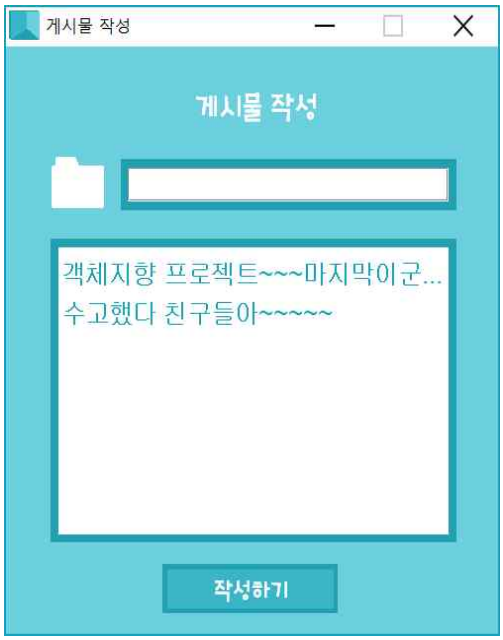
-로그인과 회원가입과 아이디 비밀번호를 찾을 수 있다.

<아이디/ 비밀번호 찾기> -아이디와 비밀번호를 찾을 수 있다. -비밀번호는 비밀번호 질문에 답을 해야 수정가능하다.

<회원가입>
-가입하고 싶은 정보로 회원가입을 하며 작성하기 버튼을 누른다.
(아이디와 비밀번호는 4~9자리 영자와 숫자여야하고 이메일은 '@', '.'를 꼭 넣어줘야한다.



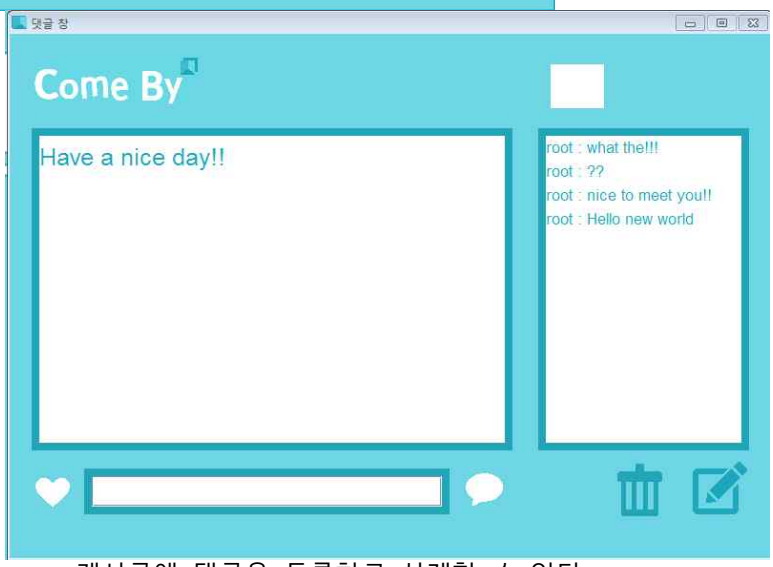
<홈 화면>
-홈 화면에서는 타임라인, 다른 계쩡 검색, 프로필, 프로필 창, 미니룸, 채팅을 볼 수 있다.



<게시물 작성>
-등록하고 싶은 글을 입력한다.

<게시물 이동>
-게시물을 이동시켜서 볼 수 있다.

<프로필 설정>
-프로필을 수정/정보 등록 할 수 있다.
<댓글 창>



-게시글에 댓글을 등록하고 삭제할 수 있다.
-댓글을 작성 할 수 있다.

XII.Work distribution & Progress of Each member

이름	주요 클래스	세부 사항	라인 수
강수지	GUI_Profie GUI_Post GUI_Home GUI_Home_Other	저는 프로필 설정 창, 홈에서 보이는 프로필, 게시물 올리기, 팔로우, 다른 사용자의 계정 검색해서 이동하는 기능을 맡았습니다. 첫 번째로 Profile 설정 창에서 이메일과 아이디는 수정을 할 수 없고 텍스트를 데이터베이스에서 불러와서 정보를 보여줍니다. 그리고 프로필 사진을 파일탐색기를 통해서 프로필 사진을 설정할 수 있고 그 선택한 이미지는 프로필 설정 창에 사이즈를 맞춰서 이미지 등록하기 전에 미리 보여줍니다. 그리고 미니룸에 띄울 캐릭터를 선택할 수 있게 만들었고 프로필의 한 줄 소개를 작성 할 수 있으며 사용자의 계정 정보를 수정할 수 있습니다. 그리고 사용자 계정의 비밀번호를 현재 비밀번호를 작성해야지 바꿀 수 있고 비밀번호를 다시 작성해 이전과 비밀번호가 일치하는지 확인을 합니다. 이렇게 정보를 작성과 수정을 하지만, 현재 비밀번호와 일치해야지 프로필 정보를 수정할 수 있게 제안을 두었습니다. 두 번째로 홈 화면의 프로필 화면에서는 뉴스피드로 이동을 해도 프로필은 똑같이 보이고 프로필 기능을 합니다. 그리고 홈 화면에서는 로그아웃 버튼을 추가하여 로그아웃 버튼을 누르면 프로그램을 종료할 수 있습니다. 또한 검색 창에서는 다른 계정을 검색해서 사용자 정보를 저장하는 벡터에서 데이터베이스에 저장된 아이디들을 각각 비교하여 일치하는 아이디가 있으면 그 아이디의 계정으로 이동하는 기능을 만들었습니다. 마지막으로 홈 화면에서는 뉴스피드로 이동하는 버튼을 만들어 뉴스피드로 이동할 수 있도록 만들었습니다. 세 번째로 Post에서는 게시물을 추가하는데 원래 계획에서는 게시물로 이미지와 글을 타임라인과 뉴스피드에 올리는 것이었지만 시간 부족으로 인해 게시물 이미지 올리는 것은 하지 못했습니다. 그러나 게시글은 게시글을 작성하면 타임라인과 뉴스피드에 최신 글 순으로 올라가며 데이터베이스를 업데이트 하고 데이터베이스에 게시물이 추가되면 게시물 수가 1씩 증가합니다. 그리고 게시물 수가 4개 이상이 된다면 뒤로 가기 버튼을 눌러 나중에 작성된 게시물을 볼 수 있게 했으며 앞으로 가는 버튼을 누르면 게시물을 이동시켜 다시 최근에 쓴 글을 볼 수 있게 만들었습니다. 네 번째로 팔로우는 자신의 홈이면 팔로우를 하는 사람의 수를 보여주고 팔로우 리스트를 구현한다고 계획을 하였지만 팔로우 리스트의 기능은 구현하지 못했습니다. 하지만 팔로	1000줄

		우 수를 보여줍니다. 사용자가 다른 사람의 계정으로 들어가서 팔로우 버튼을 누르면 팔로우 버튼의 색이 바뀌고 팔로우 정보는 데이터베이스에 추가합니다. 하지만 언팔로우를 하는 경우가 있습니다. 그런 경우를 생각해서 다시 버튼을 누르면 버튼을 누르지 않았던 상태로 바뀌며 데이터베이스에 팔로우로 저장된 계정 정보를 삭제합니다. 다른 사용자의 계정의 홈에서는 뉴스피드로 이동하는 버튼이 자신의 홈으로 이동하는 기능으로 바뀌고 팔로우 버튼의 모양이 바뀌며 팔로우 수를 보여주지 않고 팔로우 기능을 할 수 있고 나머지는 홈과 동일합니다. 대신 게시물은 작성할 수 없습니다.	
김서현	Server Client	서버와 클라이언트간의 통신이 가능하고 서버와 데이터베이스를 연결해 결과적으로는 클라이언트들이 서버를 통해 데이터베이스에 값을 전달하고 받아 여러 사항들을 변경하고 명령을 수행할 수 있도록 Server와 Client클래스를 구현하였습니다. 서버 클래스에서는 클라이언트의 접속을 허용하고 로그인을 하기 전 사용자는 임시로 저장하기 위해 임시 벡터를 만듭니다. 로그인을 한 후에는 접속이 확인된 사용자를 저장하기 위해 데이터베이스에서 이름을 가져와 미리 만들어 놓은 벡터의 인덱스를 이름으로 검색해 동일한 인덱스에 접속한 사용자의 스레드 값을 저장합니다. 동일하게 사용자의 위치 정보와 사용자의 홈 화면에 접속한 다른 사용자의 정보 또한 각각의 벡터에 저장합니다. 이를 통해 접속된 클라이언트를 구분하고 각 클라이언트가 보낸 데이터를 분석해 클라이언트의 요구에 맞는 데이터를 다시 전달해주는 동작을 만들었습니다. 클라이언트가 보낸 데이터를 분석하는 동작은 manager메소드에서 이루어지며 각 데이터는 맨 처음과 두 번째 구분 기호로 구분된 flag값을 가져 이 flag를 비교하여 어떤 요구인지 구분합니다. 로그인, 회원가입, 아이디찾기, 비밀번호찾기, 정보 추가하기, 정보 갱신하기, 정보 삭제하기, 위치 정보 갱신하기, 채팅 메시지 보내기 등의 명령이 flag로 각각 구분되어있으며 그 후 동작을 수행하기 위해 추가적으로 필요한 정보도 데이터에 구분 기호로 들어갑니다. 해석된 명령은 해당 명령을 수행하기 위해 DB클래스의 메소드를 호출하거나 다시 클라이언트로 데이터를 보내는 등의 동작을 하게 됩니다. 클라이언트 클래스에도 반대로 서버로부터 온 데이터를 해석하는 manager메소드가 있습니다. 서버의 manager 메소드와 동일한 방법으로 사용되며 이 때는 GUI 클래스들의 메소드나 벡터로 정보가 저장된 Data 클래스의 메소드를 호출하게 됩니다.	660
손휘진	ChatInfo PostInfo	저는 데이터베이스와 관련된 대부분의 메소드를 작성하고 초기화면에 포함된 기능(로그인, 회원가입, 아이디/비밀번호 찾	1000

	UserInfo DB GUI_ChangePW GUI_FindIDPW GUI_FindPW GUI_Join GUI_Login	기)을 구현하는 역할을 맡았습니다. 데이터베이스는 MySQL을 사용하였고 comeby라는 이름을 가진 데이터베이스로 user, posting, chatting의 테이블이 있습니다. user테이블은 사용자의 정보(port, id, pw, email, name, pwQuestion, pwAnswer, introduce, animal, followerNum, postNum, followee)를 저장하는 테이블입니다. port는 채팅시스템을 위해 만들어놓은 임시 필드로 기본값은 10000이고 user을 추가할 때마다 자동으로 1씩 증가되도록 하였습니다. id는 사용자의 id를 저장하는 필드이고 pw는 비밀번호는 저장하는 필드로 password()함수를 통해 암호화를 하여 저장합니다. email, name은 각각 이메일과 이름이고 pwQuestion은 본인확인 질문, pwAnswer은 본인확인 질문에 대한 답을 나타냅니다. introduce는 자기소개, animal은 채팅방에서 표시할 사용자의 캐릭터입니다. followerNum은 자신을 팔로우한 사람들의 수, postNum은 자신의 총 게시물 수, followee는 자신이 팔로우한 유저들을 구분기호 $를 통해 저장하는 필드입니다. post테이블은 게시물의 정보(date, id, text, loc, c1, c2, liker)를 저장하는 테이블입니다. 필드는 각각 게시물 작성 날짜, id, 게시물의 내용, 사진의 경로, 게시물 코드, 댓글 코드, 좋아요 누른 사람들을 구분기호($)를 통해 나눈 String입니다. post테이블에는 게시물의 정보는 물론 댓글의 정보도 저장을 합니다. 게시물의 경우 c1에는 게시물코드, c2에는 "NULL"을 저장하며 댓글의 경우 c1에는 댓글 쓴 게시물코드, c2에는 댓글코드를 저장하도록 하였습니다. 각각의 코드는 YY, MM, DD, 작성자 ID, 그 날짜에 작성된 순서를 이어서 저장하도록 하였습니다. chatting테이블은 채팅 정보(date, receiver, sender, text)를 저장하는 테이블로 각각 보낸 날짜, 받는 사람, 보낸 사람, 채팅내용을 저장합니다. 유저는 프로그램을 실행시킬 때 DB에서 이 테이블들의 모든 정보들을 불러와서 DB클래스의 userInfo_vector, postInfo_vector, chatInfo_vector 벡터에 저장을 합니다. 그 뒤로 로그인, 회원가입, 게시물 작성 등 대부분의 작업을 벡터를 통해 하며 DB에 저장해야하는 경우 DB클래스에 작성된 메소드들을 사용하도록 하였습니다. DB클래스에는 데이터베이스에 연결하는 connectDB()를 포함하여 데이터베이스에 쿼리문을 보내는 inputQuery(String), 벡터에서 값을 검색하는 search메소드 등 데이터 관리에 필요한 메소드들이 포함되어 있습니다.	

		초기화면(GUI_Login)에는 로그인 기능, 회원가입 기능, 아이디/비밀번호 찾기 기능을 넣었습니다. 회원가입의 경우 정보를 입력받아서 적절한 형식인지 확인한 후 데이터베이스에 입력하도록 하였습니다. 아이디/비밀번호 찾기는 아이디를 찾을 때에는 이름, 이메일을 입력받고 비밀번호를 찾을 때에는 아이디, 이메일, 본인확인 질문에 대한 답을 입력받고 비밀번호를 재설정할 수 있도록 하였습니다. 로그인의 경우 아이디와 비밀번호를 입력받은 후 아이디가 일치하는 경우 데이터베이스에 연결한 후 password() 메소드를 호출해서 입력받은 비밀번호를 암호화시키고 암호화된 정보가 기존 데이터베이스에 입력된 패스워드와 일치하는 경우 로그인을 성공시키도록 하였습니다. 현재 제 파트인 데이터베이스와 관련된 모든 사항과 초기화면에 들어가는 모든 기능은 구현된 상태입니다.	
우영주	GUI_Home_Other GUI_Newsfeed GUI_Comment	저는 프로젝트의 전반적인 스윙작업을 맡았으며 뉴스피드와 댓글 창, 그리고 강수지 학생과 함께 다른 유저의 홈 화면, 각 게시물을 댓글창과 이어서 출력하고 댓글을 DB에 올리는 기능, 게시글을 삭제하는 기능, 이전 게시물을 볼 수 있는 기능, 좋아요 기능을 맡았습니다. 뉴스피드의 기능에서는 게시물 작성 버튼, 좋아요 버튼, 댓글버튼, 최신순의 게시물 4개, 프로필 설정, 홈화면 이동 버튼, 로그아웃 버튼, 팔로우 리스트 버튼이 있습니다. 게시글 작성 버튼은 누를 시에는 게시글 작성 창으로 넘어가게 되는데 여기서 게시물은 텍스트형식으로만 저장이 됩니다. 포스팅을 하게 되면 게시글의 작성시간, 텍스트, 아이디, 게시글 코드, 댓글코드들이 DB에 저장되고 홈 화면 뉴스피드 게시글 박스에 갱신됩니다. 여기서 갱신된 게시글 박스에는 좋아요 버튼과 댓글버튼이 있는데, 좋아요를 누르면 좋아요 메소드가 호출되어 아이콘의 모양이 바뀌고 DB에는 좋아요를 누른 사람의 ID가 저장 되어야 되는데 현재 그렇게는 구현 하지 못하였고 좋아요 버튼 클릭만 가능하게 구현 되어 있습니다. 그리고 댓글 버튼을 누를 시에는 버튼위의 게시물 박스에 있는 게시글의 내용이 댓글 창의 게시글 박스에 불러오게 되고 댓글을 작성하게 되면 작성된 댓글이 댓글 박스에 갱신됩니다. 댓글박스는 새로 쓸 때마다 새로 초기화 되고 이전 내용에 추가된 댓글이 모두 한번에 올라가게 됩니다. 이렇게 작성된 댓글 또한 DB의 댓글 코드에 저장됩니다. 이때 DB에서는 텍스트에 댓글 코드가 저장됩니다. 이 댓글 창에서는 또한 게시물 수정 버튼과 삭제버튼이 둘다 있는데 현재 게시물 수정 버튼은 수정은 되지 않고 현재 게시물 작성으로 넘어가게 되어있습니다. 그리고 삭제버튼은 DB에 있는 게시물 삭제 메소드를 이용하여 그 게시물에 맞	1375

		는 게시물 코드를 삭제하게 하였습니다. 그리고 뉴스피드에서는 홈 화면과 마찬가지로 프로필 창을 띄울 수 있는 버튼, 다른 유저를 검새가고 유저의 홈 화면으로 가는 버튼, 자신의 홈 화면을 가는 버튼과 로그아웃 버튼이 있습니다. 다른 유저의 홈 화면은 자신의 홈 화면의 기능과 대부분이 동일하고 다른 점은 게시물 박스에서 홈 화면 주인의 게시물만 나온다는 것과 게시물 작성 버튼이 없다는 것이 있습니다.	