Mid-term

Timetable helper website for students at Hongik University

B811222 신동원

B811228 최성혁

B735163 도은채

Timetable helper website

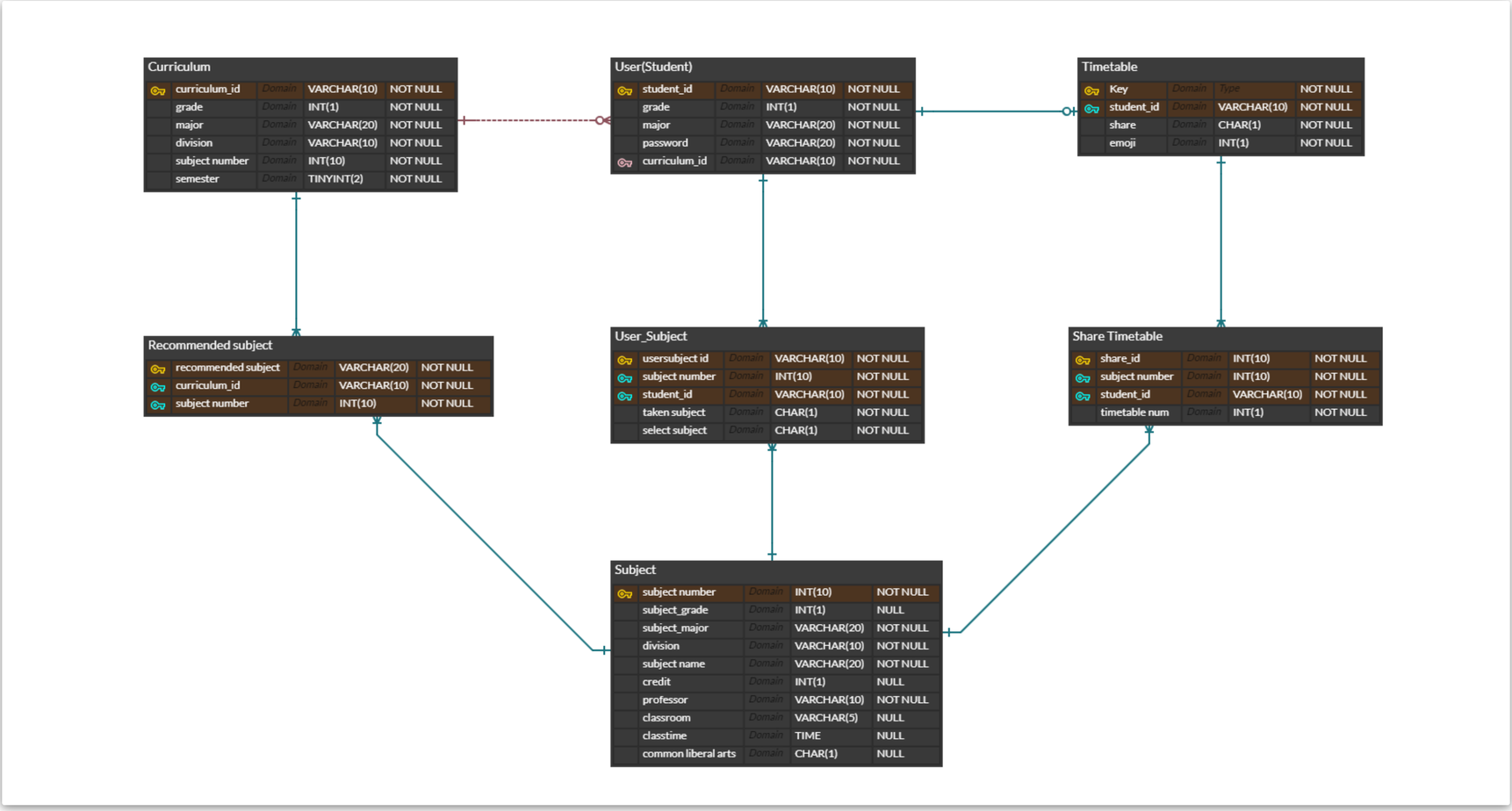
Advice message box

Systemic lecture listing system

Keyword recommendation system

Design

1 Database Design



Design

1 Database Design

Recommendation Algorithm

Using mapping table

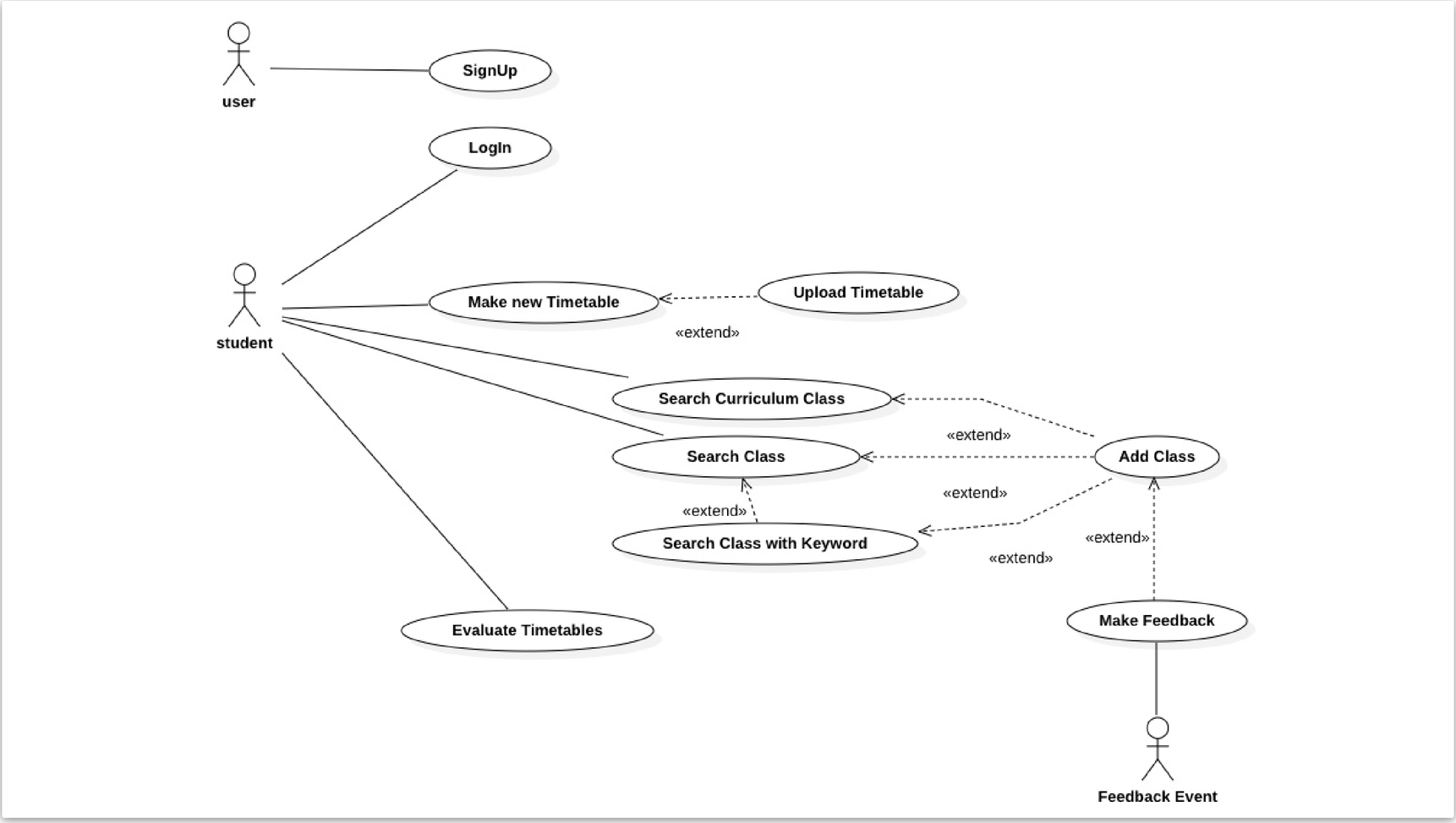
Design

2 Requirement List

1	User signs up by enteringn their school ID, password, major, and subject already taken.	Sign Up
2	Student logs in using their school ID, password.	Log In
3	Student selects a subject from compulsory class list and make new timetable with the selected classes.	Make New Timetable
4	Student searches for recommended classes in the curriculum.	Search Curriculum Class
5	Student searches for a class with class name, professor, area.	Search Class
6	Student searches for classes with selecting keywords.	Search Class with Keyword
7	Student clicks class to add the class into her timetable.	Add Class
8	Student uploads timetable to shared board.	Upload Timetable
9	Student evaluate timetables in the board with emoticon.	Evaluate Timetables
10	Each time a student adds or deletes a class, feedback event provide proper feedback.	Make Feedback

Design

3 Use Case Diagram



Design

4 Use Case Description

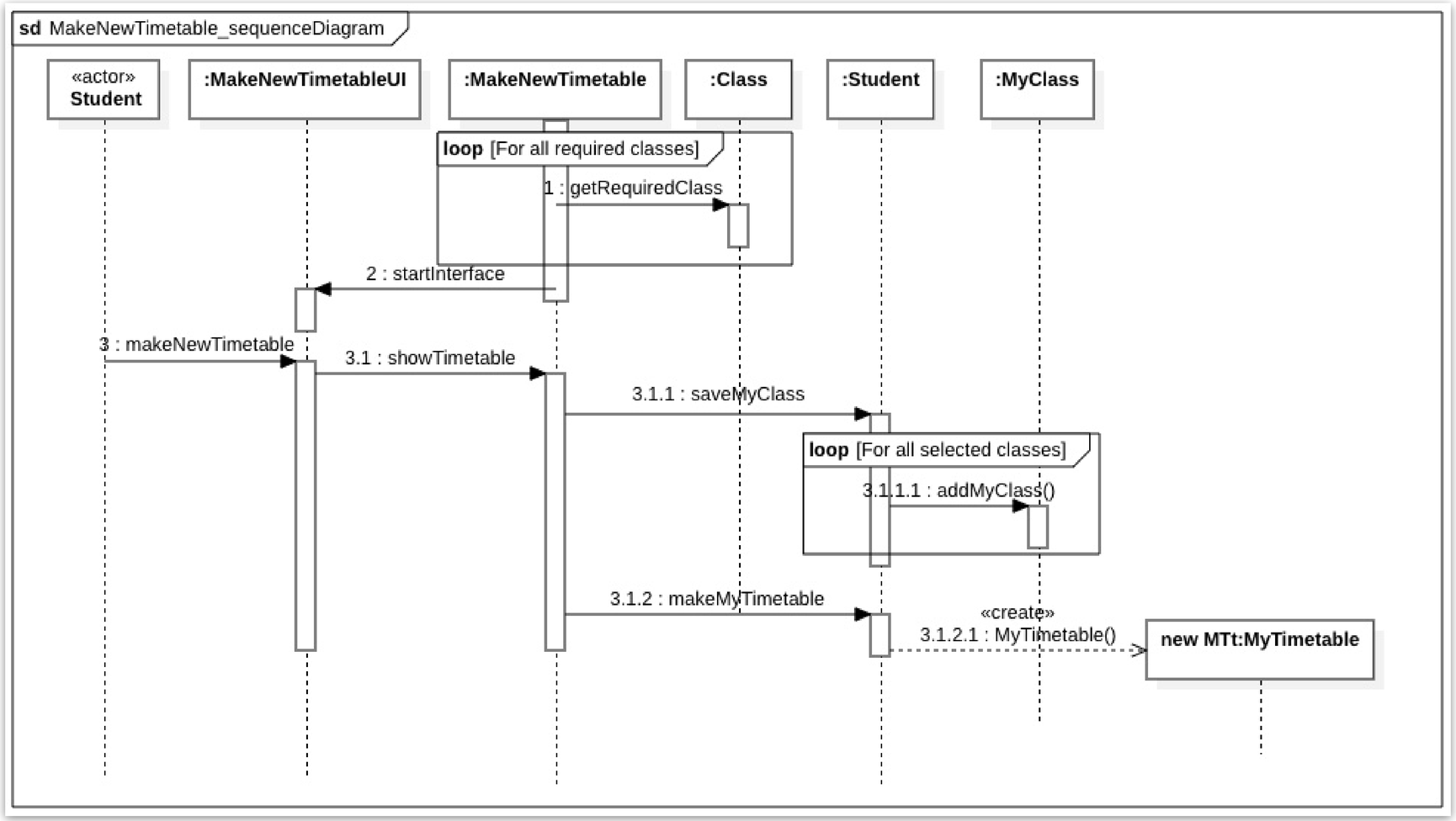
Use Case Make new Timetable

Actors	Student
Pre-Conditions	Student logged in and does not have timetable for this semester.
Post-Conditions	Student makes timetable with selected required classes.

Actor Action	System Response
	1. Lists all required classes for the student's major except those already taken.
2. The actor selects desired classes and click done.	3. Show page of new timetable with the selected classes.

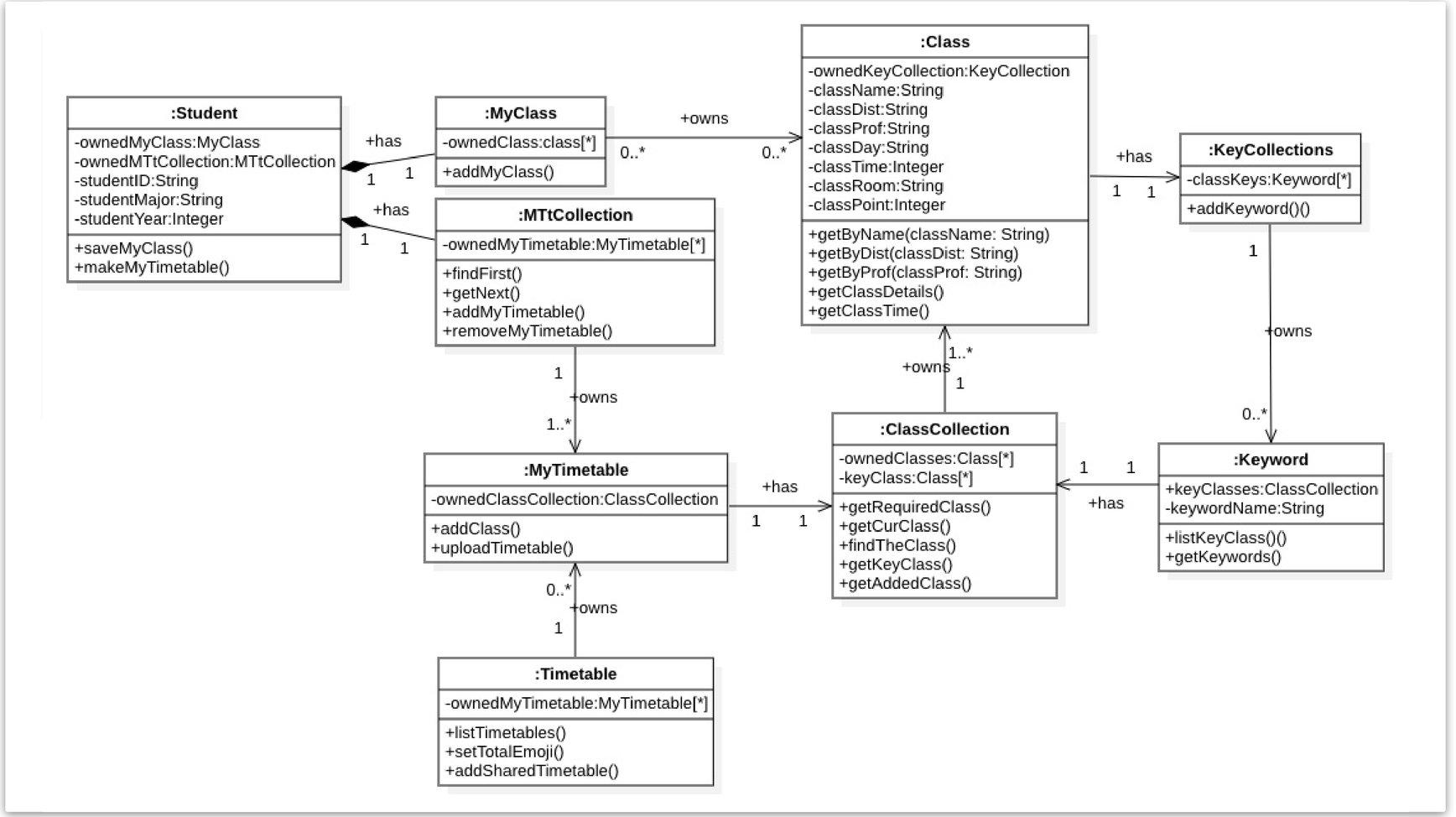
Design

5 Sequence Diagram



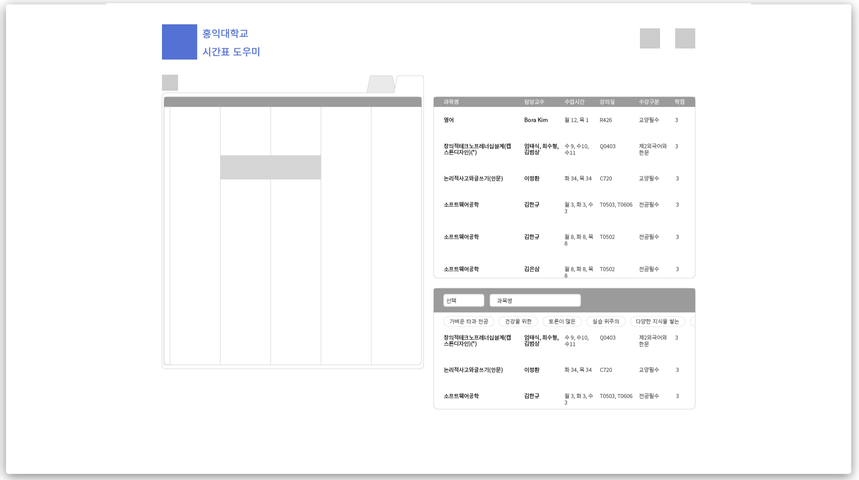
Design

6 Class Diagram



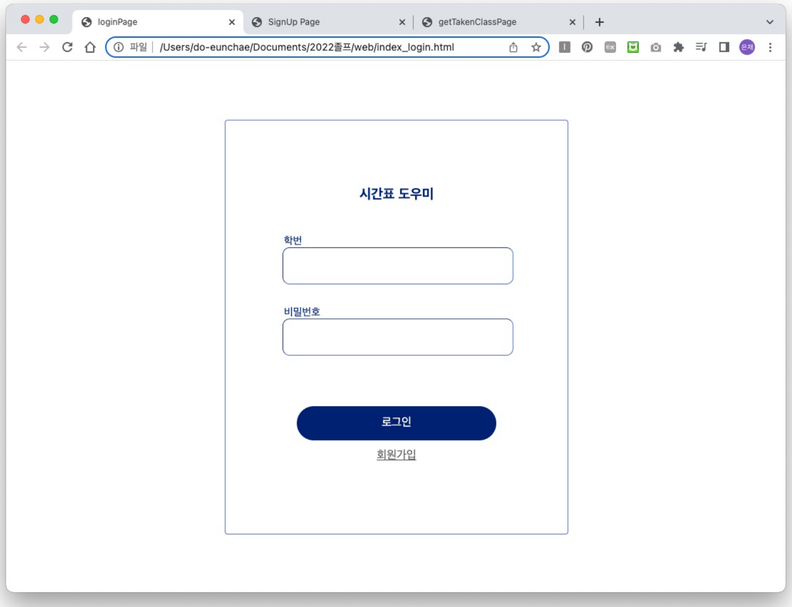
Prototyping

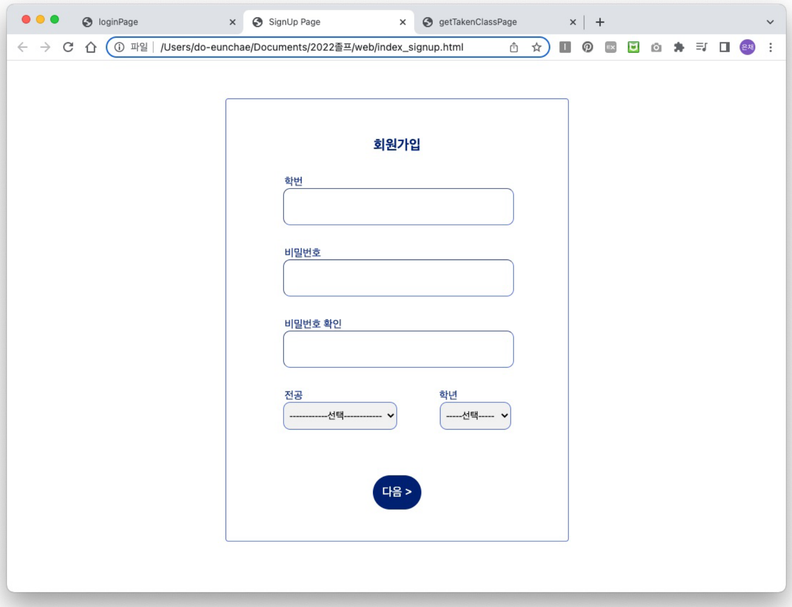
1 Screen Design



Prototyping

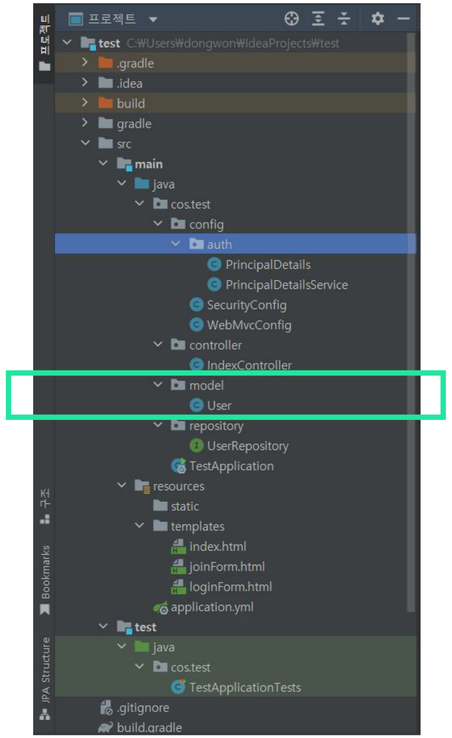
1 Login Prototype





Prototyping

2 Login Test Case

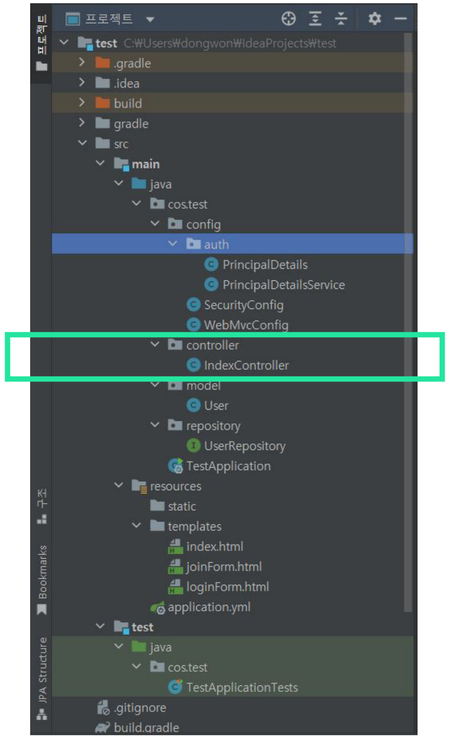


model > User

```java
package cos.test.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.sql.Timestamp;


@Data //getter setter
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class User {
    @Id//primary key
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userId;
    private String username;
    private String password;
    private String role;//ROLE_USER, ROLE_ADMIN

    @CreationTimestamp
    private Timestamp createDate;
}
```

Prototyping

2 Login Test Case

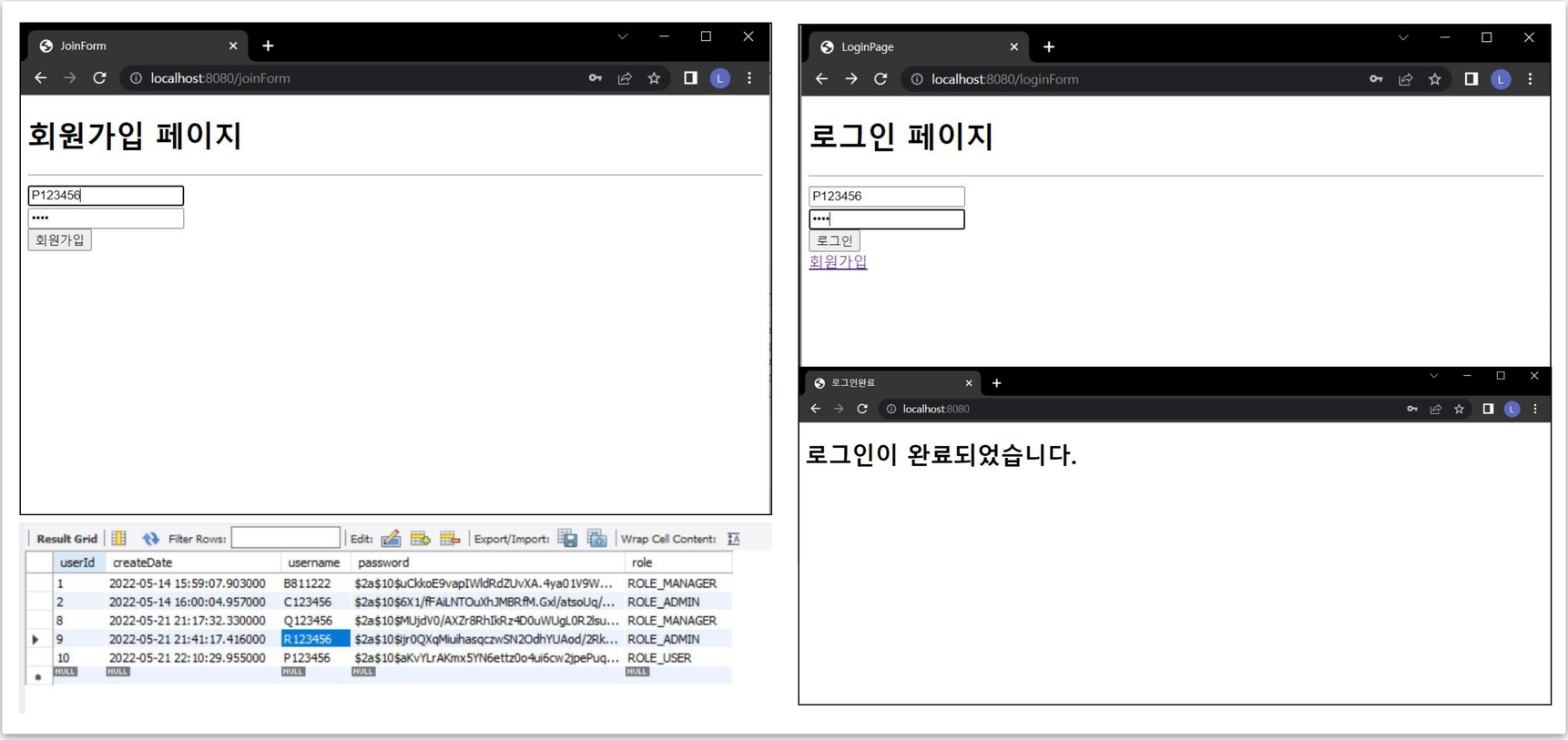


Controller

```
36
37      // 시큐리티 설정
38      @GetMapping("/user")
39      public @ResponseBody
40      String user() { return "user"; }
43      // 시큐리티 설정
44      @GetMapping("/admin")
45      public @ResponseBody
46      String admin() { return "admin"; }
49      // 시큐리티 설정
50      @GetMapping("/manager")
51      public @ResponseBody
52      String manager() { return "manager"; }
55      // Spring Security가 해당 주소를 받음
56      @GetMapping("/loginForm")
57      public String loginForm() { return "loginForm"; }
60      // 시큐리티 설정
61      @GetMapping("/joinForm")
62      public String joinForm() { return "joinForm"; }
65      // 시큐리티 설정
66      @PostMapping("/join")
67      public String join(User user) {
68          System.out.println(user);
69          user.setRole("ROLE_USER");
70          String rawPassword = user.getPassword();
71          String encPassword = bCryptPasswordEncoder.encode(rawPassword);
72          user.setPassword(encPassword);// 인코딩 완료해서 setPassword
73          userRepository.save(user); //회원가입이 잘되지만 시큐리티로는 로그인을 할 수없다 이유는 패스워드 암호화가 되지 않았기 때문
74          return "redirect:/loginForm"; //redirect:를 붙이면 /loginForm함수를 다시 불러온다.
75      }
76
```

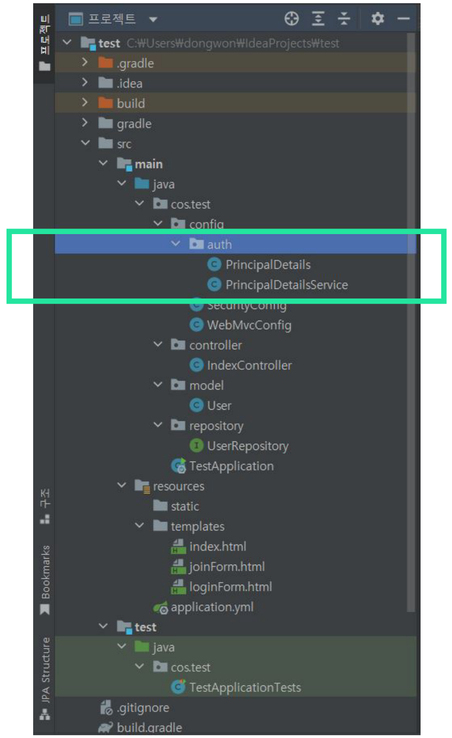
Prototyping

2 Login Test Case



Prototyping

2 Login Test Case

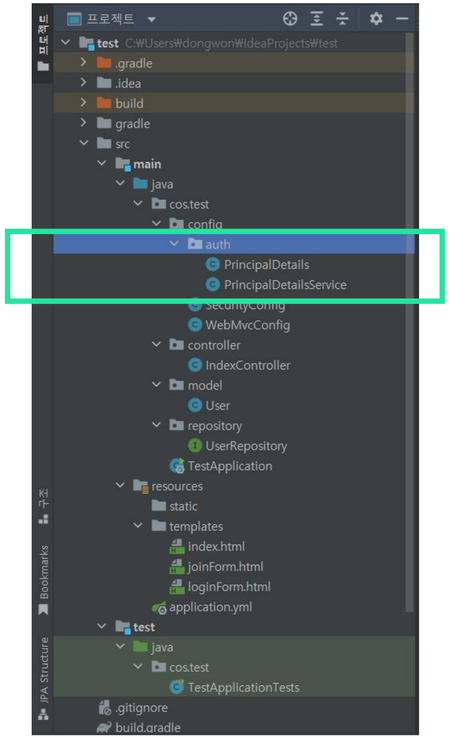


UserDetails

```java
public class PrincipalDetails implements UserDetails {

    // 우리 유저정보는 User객체가 들고있다.
    private User user;
    public PrincipalDetails(User user) { this.user=user; }
    // 해당 유저의 권한을 리턴하는 곳
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        Collection<GrantedAuthority> collect = new ArrayList<>();
        collect.add(new GrantedAuthority() {
            @Override
            public String getAuthority() { return user.getRole(); }
        });
        return collect;
    }
    @Override
    public String getPassword() { return user.getPassword(); }
    @Override
    public String getUsername() { return user.getUsername(); }
    @Override
    public boolean isAccountNonExpired() { return true; }
    @Override
    public boolean isAccountNonLocked() { return true; }
    @Override
    public boolean isCredentialsNonExpired() { return true; }
    @Override
    public boolean isEnabled() { //계정이 활성화
        // 우리사이트에서 1년동안 회원이 로그인을 안하면 휴면 계정으로 하기로 함
        // 현재시간 - 로그인 시간 -> 1년을 초과하면 return false; 이런식으로 구현을 하면 된다.
        return true;
    }
}
```

Prototyping

2 Login Test Case



UserDetailsService

```java
package cos.test.config.auth;

import cos.test.model.User;
import cos.test.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

// 시큐리티 설정에서 loginProcessUrl("/login")으로 걸어놨기 때문에
// /login요청이 오면 자동으로 UserDetailsService 타입으로 IoC되어 있는 loadUserByUsername 함수가 실행

@Service // PrincipalDetailsService가 IoC에 등록이 된다.
public class PrincipalDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;
    // 시큐리티 세션 -> Authentication -> UserDetails(PrincipalDetails)타입

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User userEntity = userRepository.findByUsername(username);
        if (userEntity != null) {
            return new PrincipalDetails(userEntity);
        }
        return null;
    }
}
```

Schedule

MARCH	APRIL	MAY	JUNE
Ideation	Design program architecture	and data architecture	
		Prototyping page layout and test case for sign-up and log-in	Developing trial performance

JULY	AUGUST	SEPTEMBER	OCTOBER
Page layout design with HTML, CSS Algorithm for feedback event	Develop interaction for recommendation system Develop pages for login, recommendation, timetable, feedback	Adapting interactions for user Develop additional pages	Efficiency improvements