

Mid-term

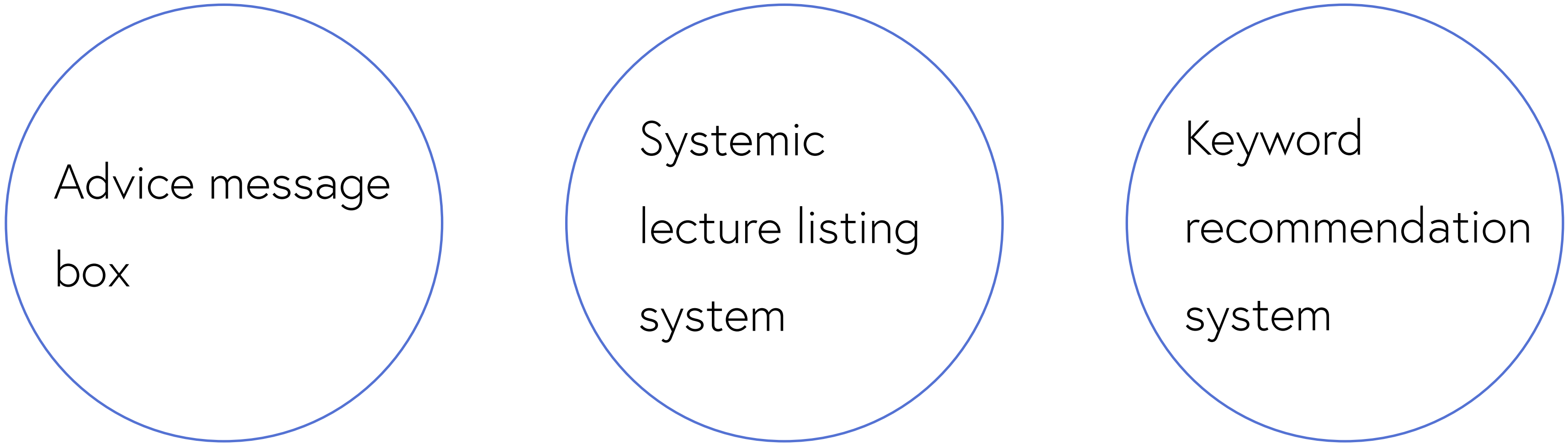
**Timetable helper website
for students at Hongik University**

B811222 신동원

B811228 최성혁

B735163 도은채

Timetable helper website



The diagram consists of three blue-outlined circles arranged horizontally. Each circle contains text describing a component of the website. The first circle on the left contains the text 'Advice message box'. The middle circle contains the text 'Systemic lecture listing system'. The third circle on the right contains the text 'Keyword recommendation system'.

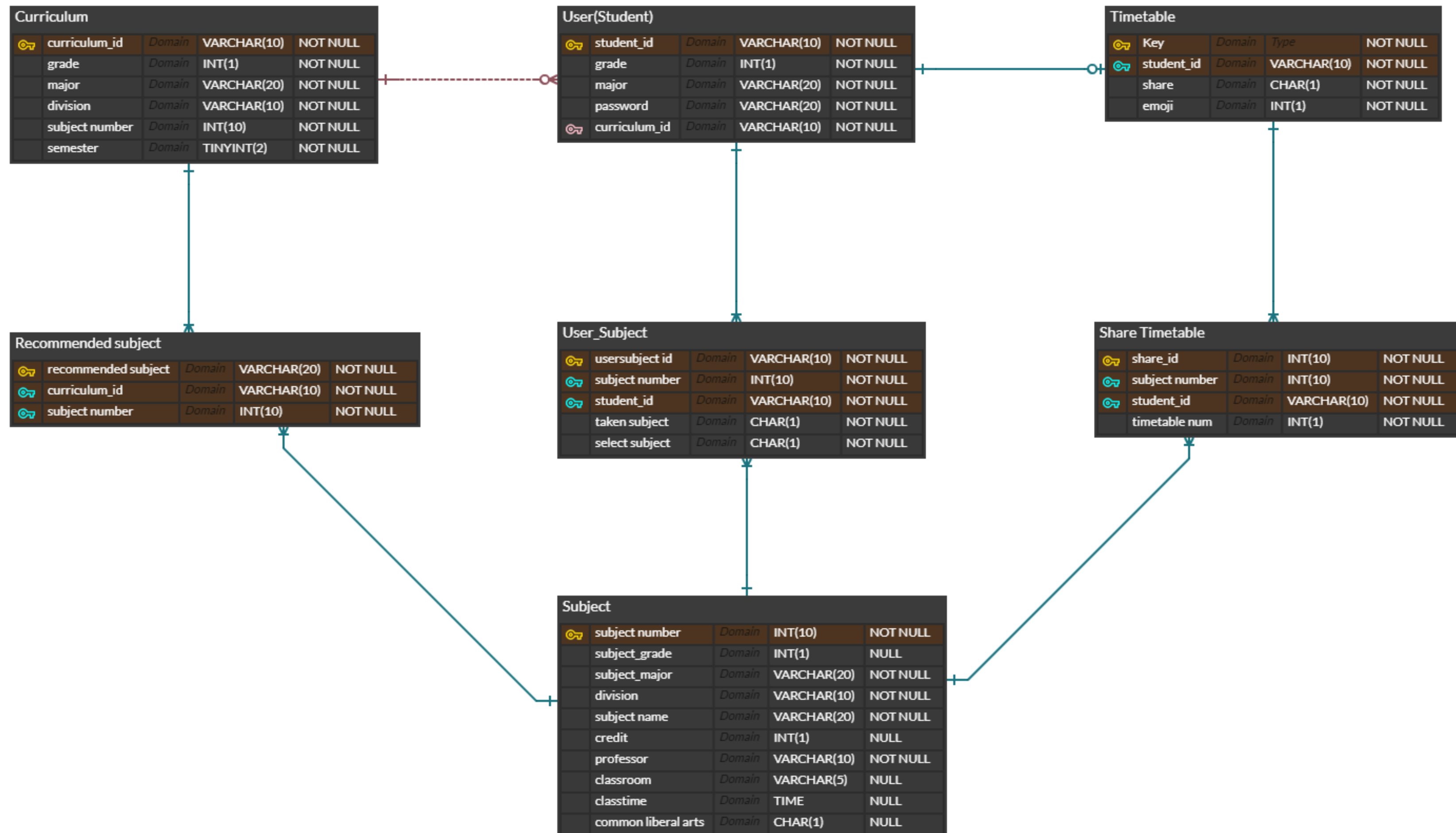
Advice message
box

Systemic
lecture listing
system

Keyword
recommendation
system

Design

1 Database Design

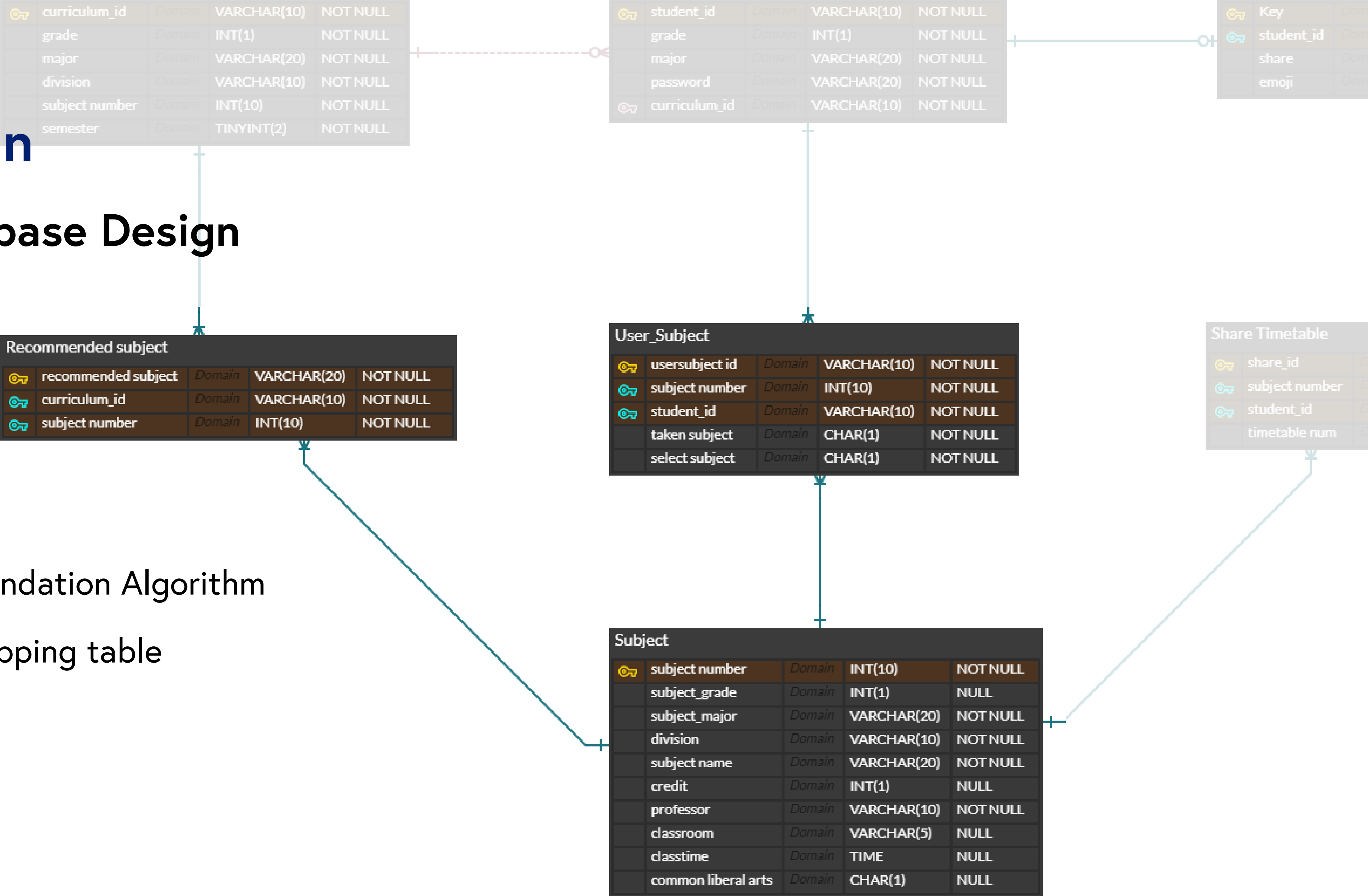


Design

1 Database Design

Recommendation Algorithm

Using mapping table



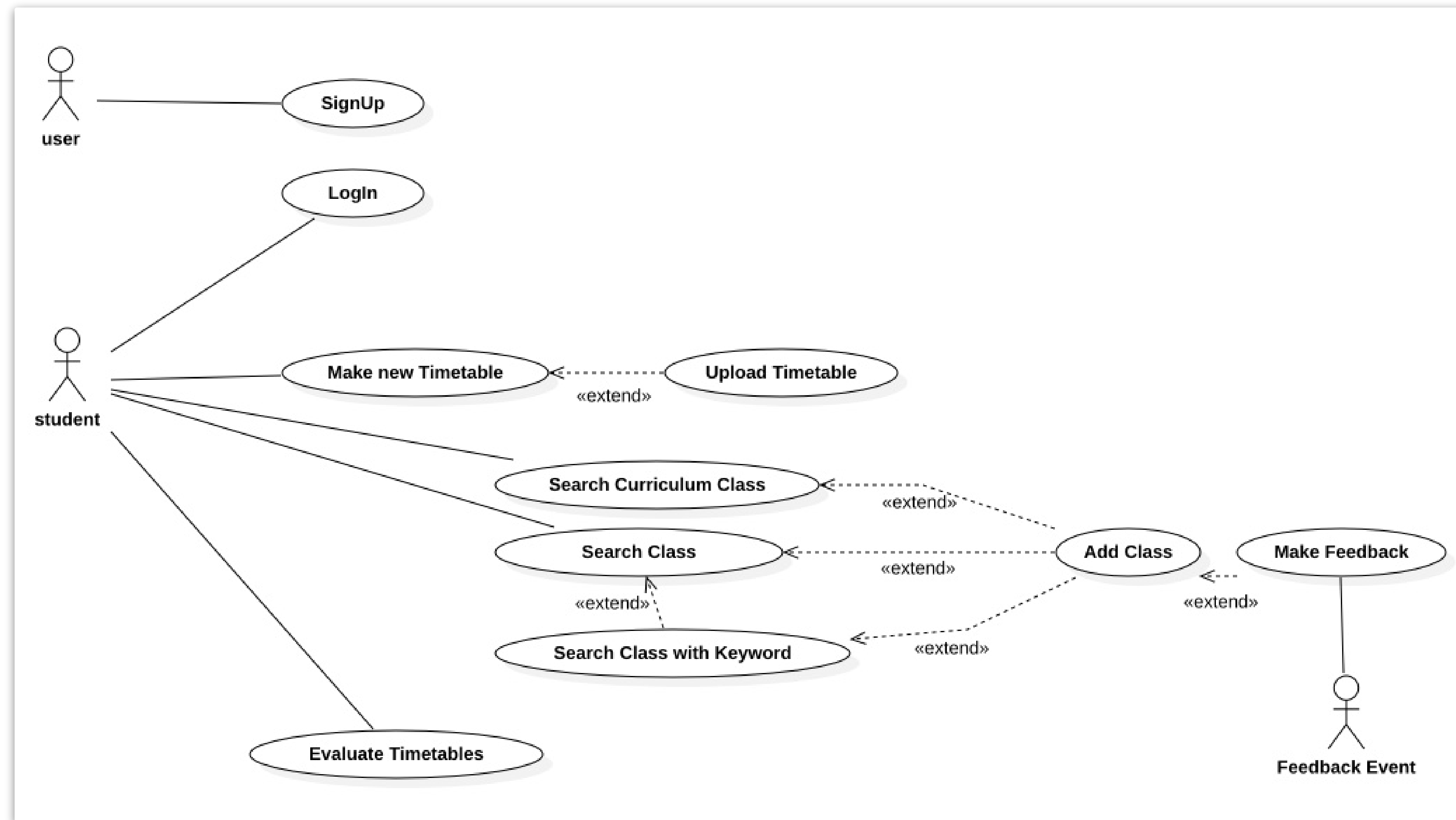
Design

2 Requirement List

- | | | |
|----|--|---------------------------|
| 1 | User signs up by enteringn their school ID, password, major, and subject al-ready 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

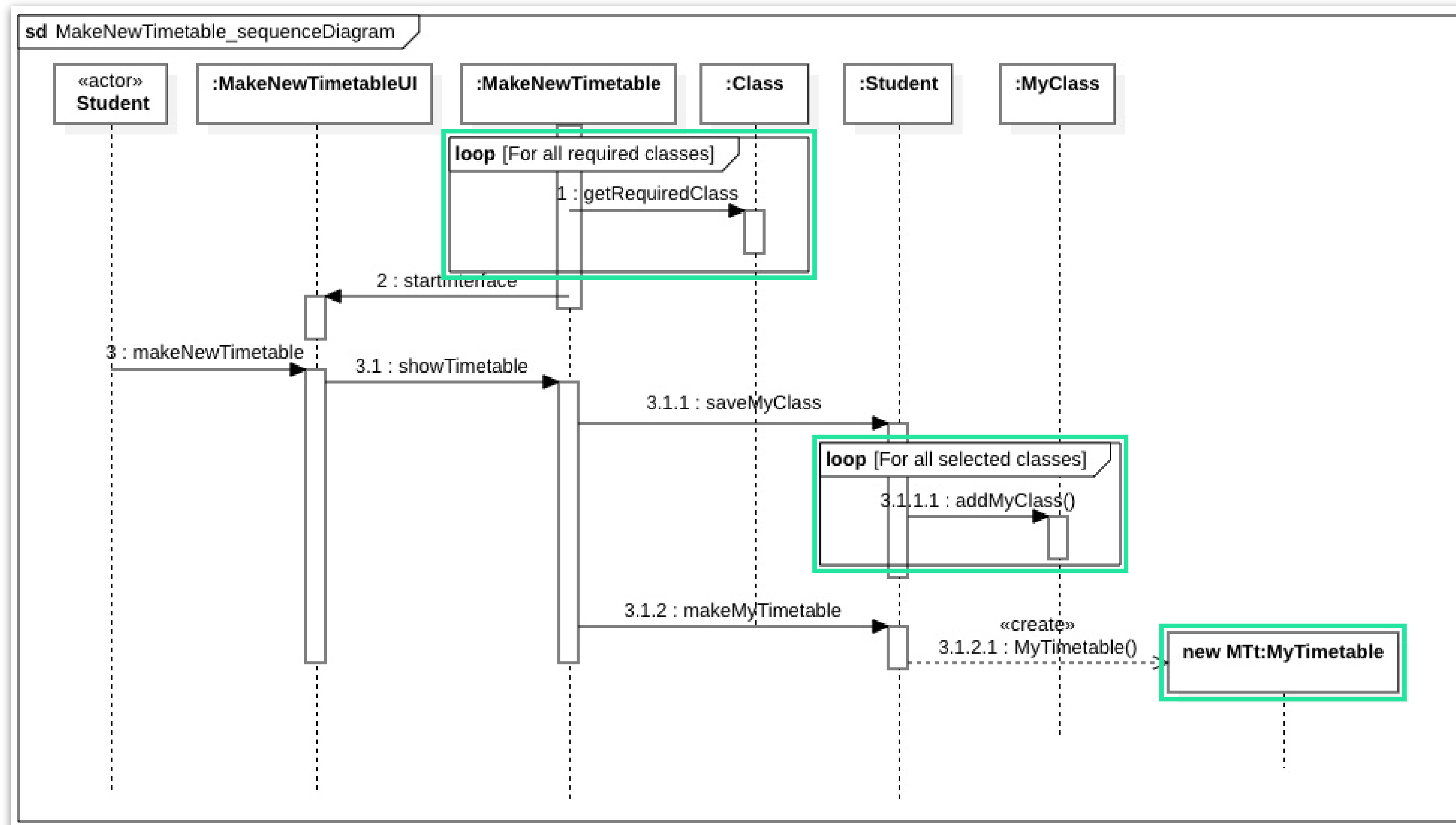
2. The actor selects desired classes and click done.

System Response

1. Lists all required classes for the student's major except those already taken.
3. Show page of new timetable with the selected classes.

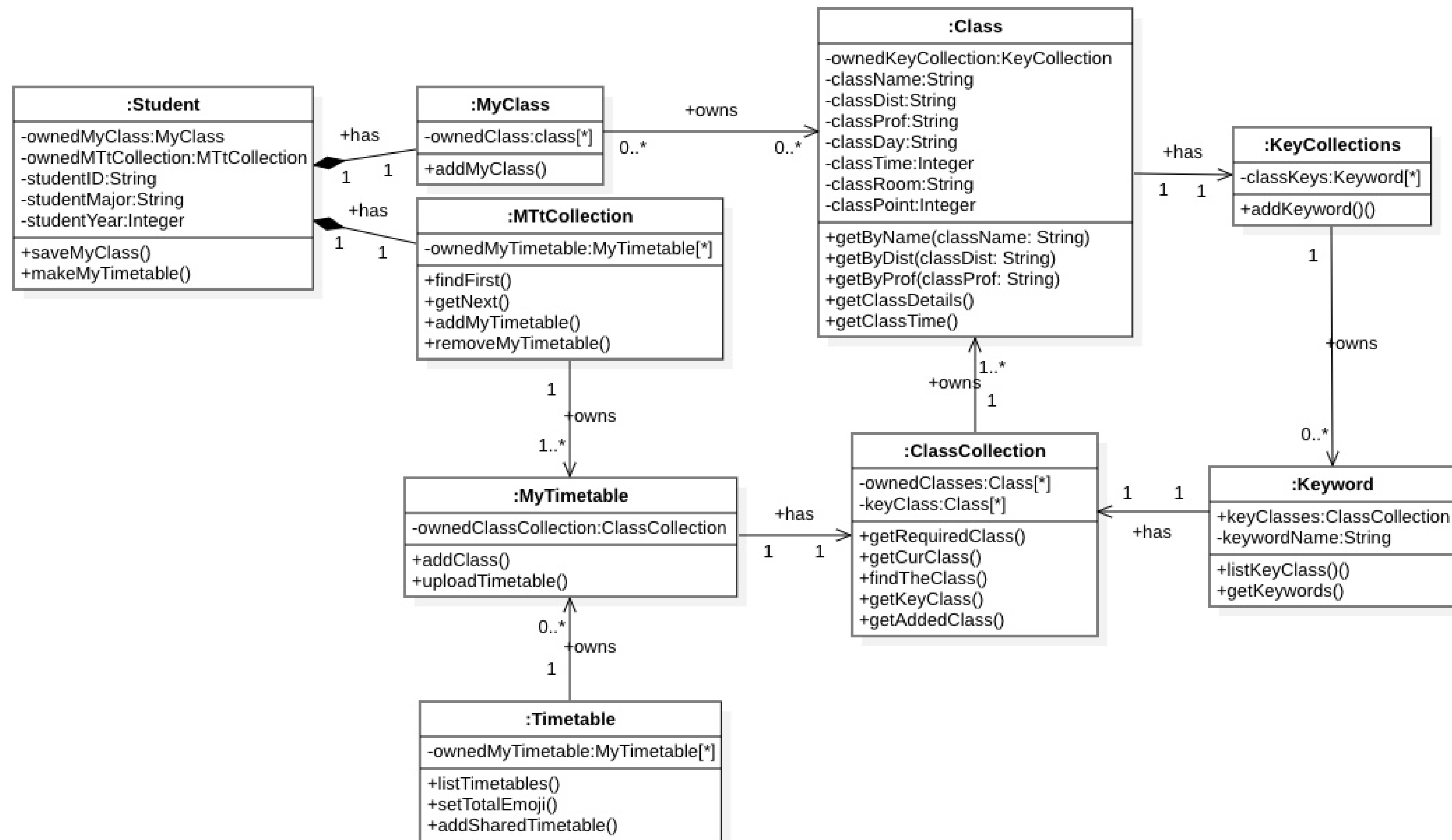
Design

5 Sequence Diagram



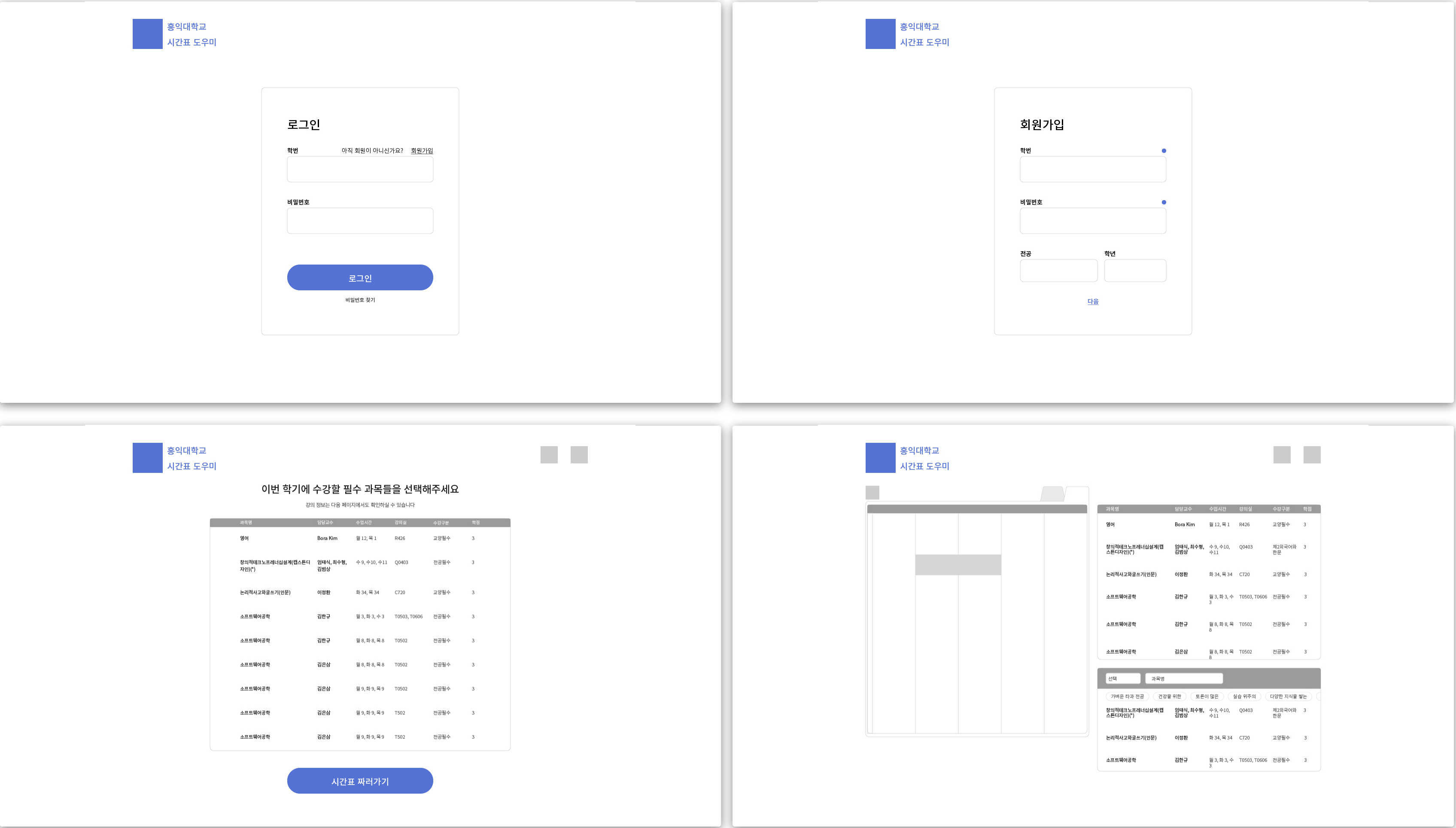
Design

6 Class Diagram



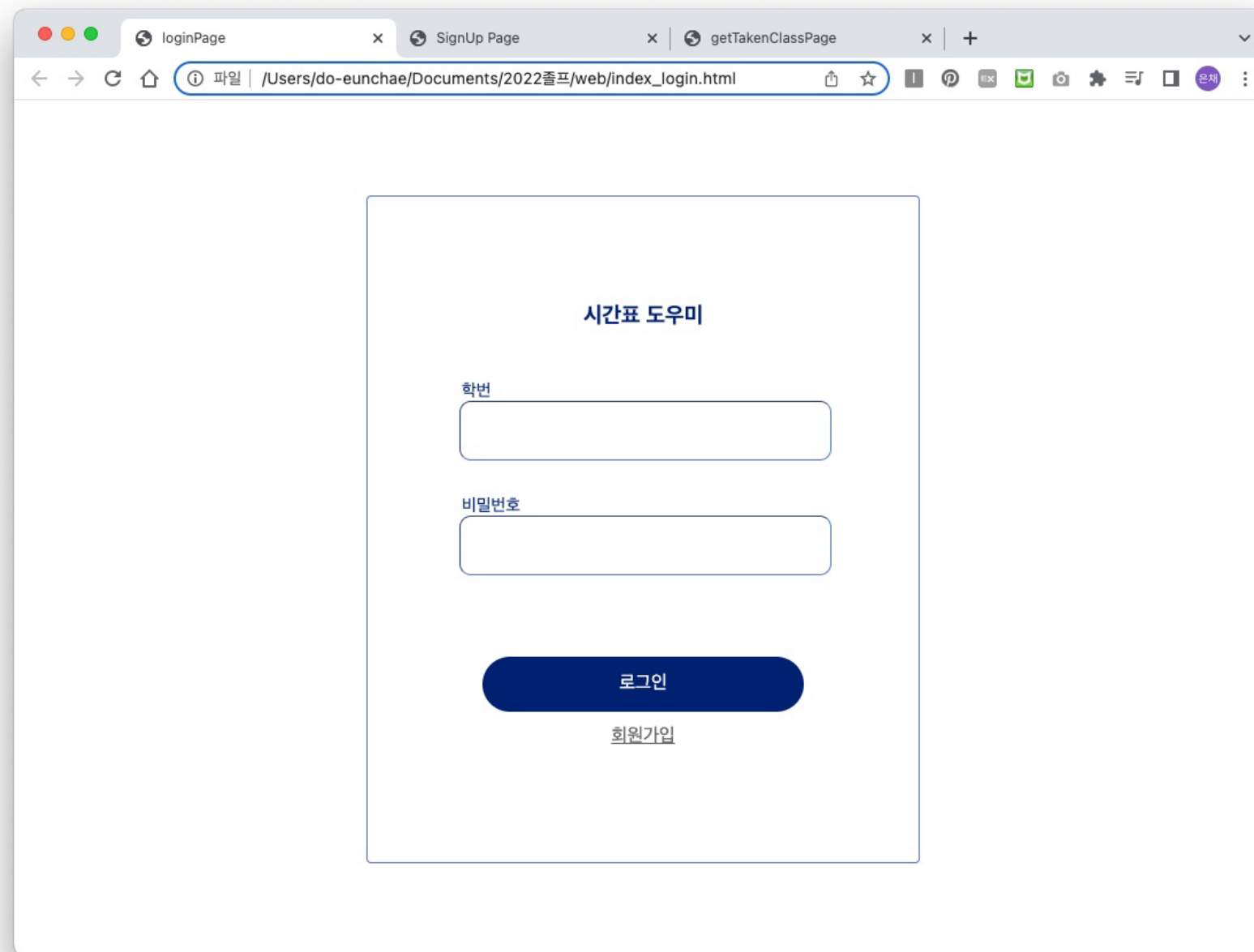
Prototyping

1 Screen Design



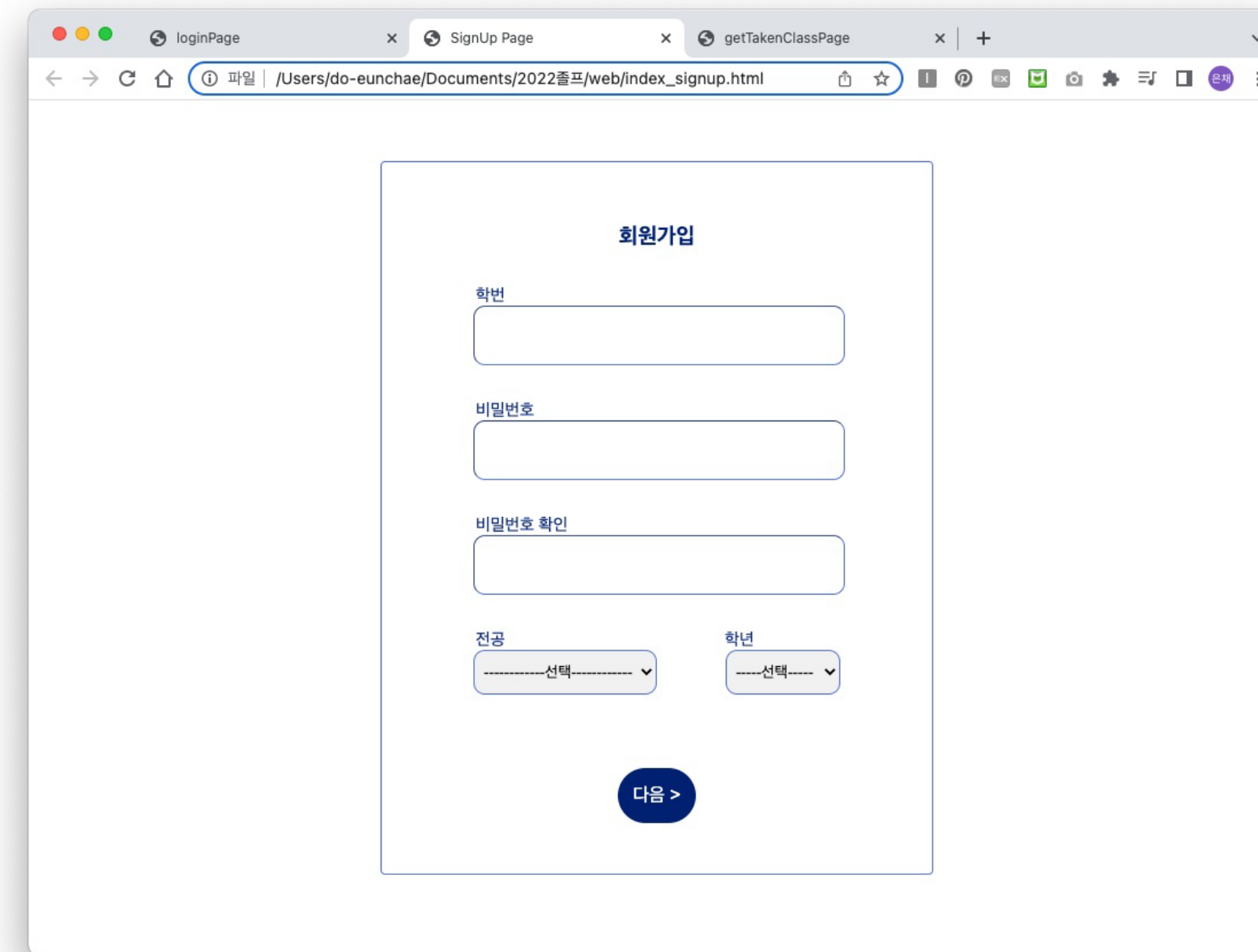
Prototyping

1 Login Prototype



A browser window showing a login page prototype. The browser tabs include 'loginPage', 'SignUp Page', and 'getTakenClassPage'. The address bar shows the file path: '파일 | /Users/do-eunchae/Documents/2022졸프/web/index_login.html'. The page content is centered and contains the following elements:

- Title: 시간표 도우미
- Form fields: 학번 (Student ID) and 비밀번호 (Password).
- Buttons: 로그인 (Login) and 회원가입 (Sign Up).

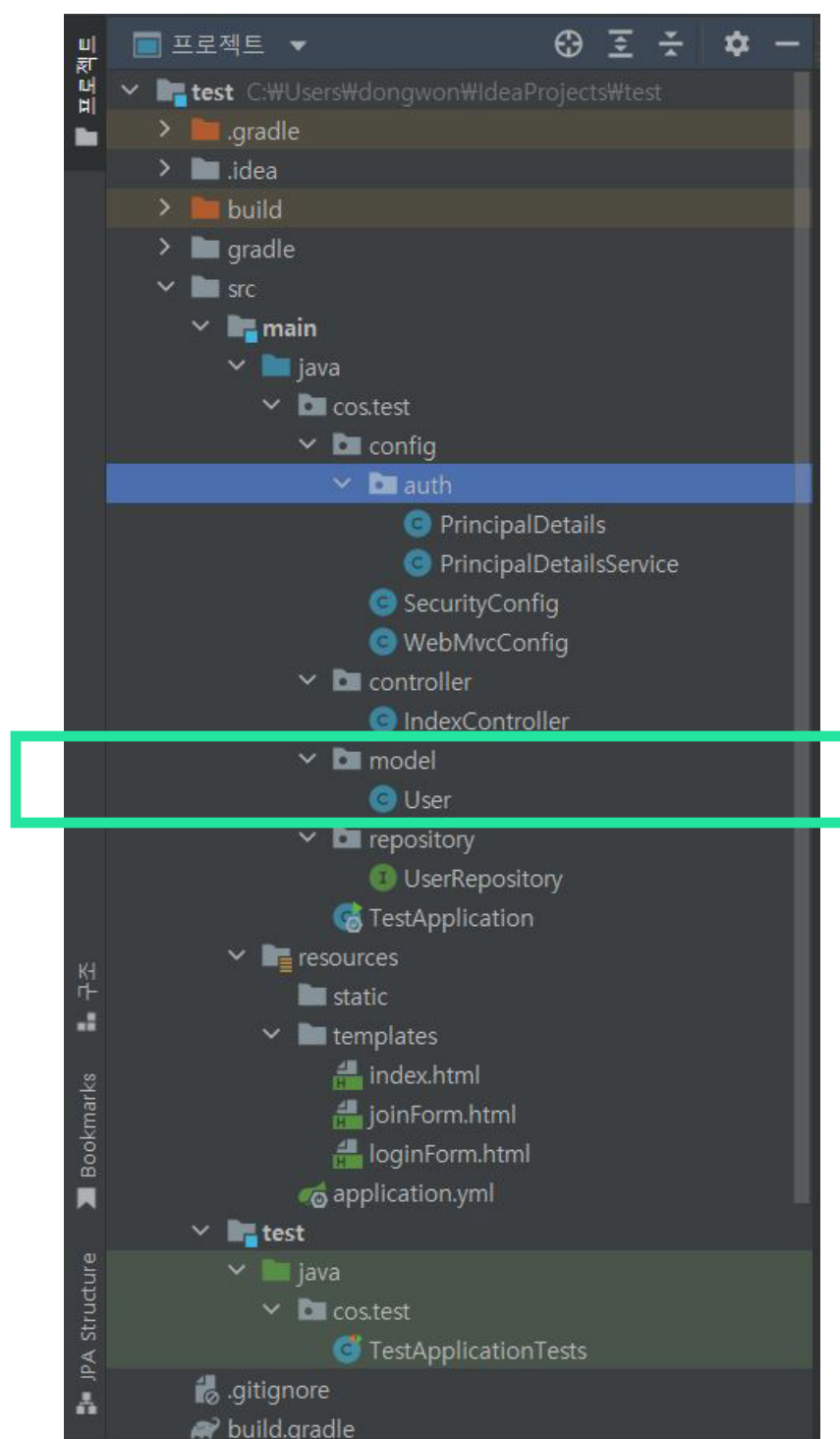


A browser window showing a sign up page prototype. The browser tabs include 'loginPage', 'SignUp Page', and 'getTakenClassPage'. The address bar shows the file path: '파일 | /Users/do-eunchae/Documents/2022졸프/web/index_signup.html'. The page content is centered and contains the following elements:

- Title: 회원가입 (Sign Up)
- Form fields: 학번 (Student ID), 비밀번호 (Password), and 비밀번호 확인 (Confirm Password).
- Form fields: 전공 (Major) and 학년 (Year) dropdown menus.
- Button: 다음 > (Next).

Prototyping

2 Login Test Case

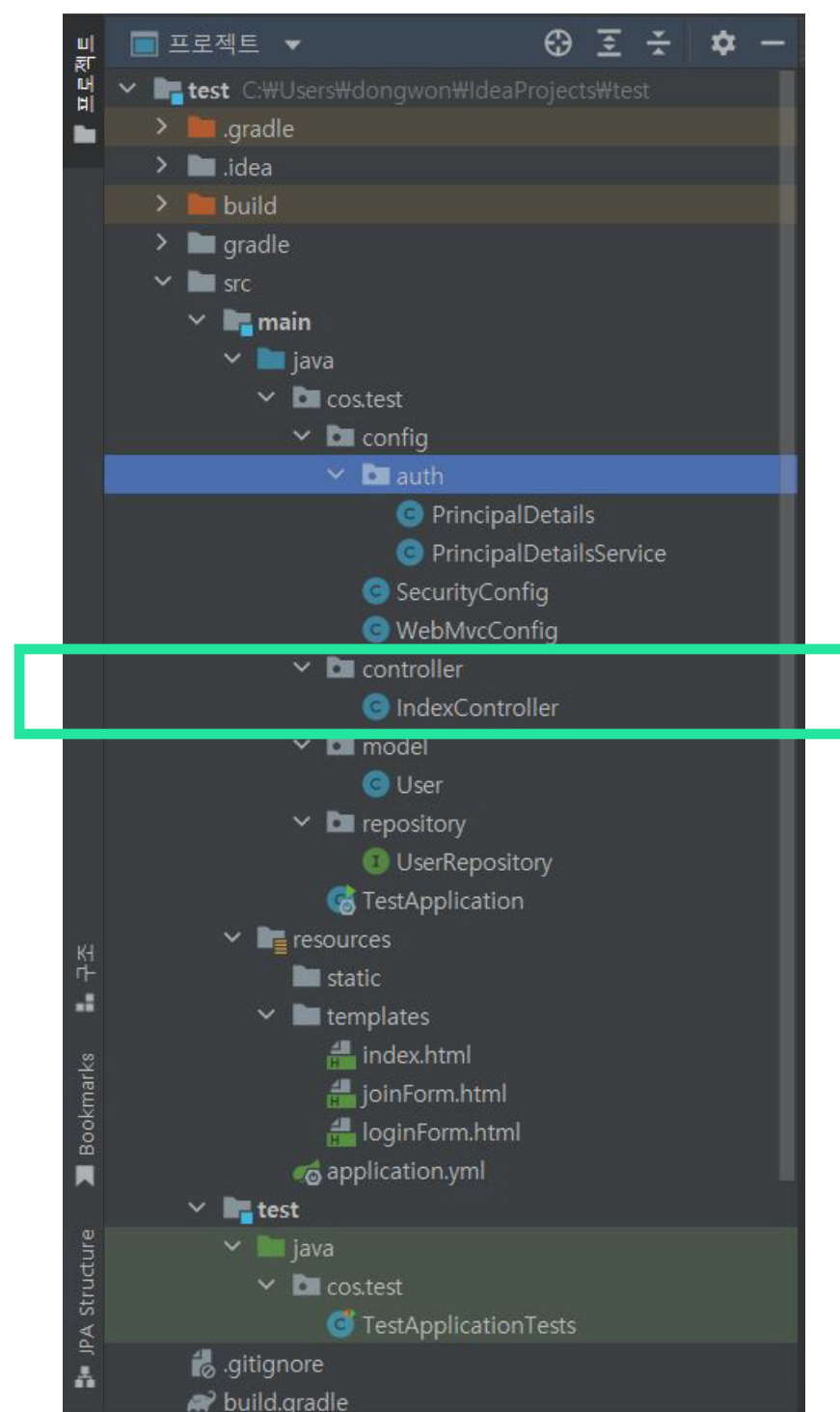


model > User

```
1 package cos.test.model;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 import org.hibernate.annotations.CreationTimestamp;
7
8 import javax.persistence.Entity;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import java.sql.Timestamp;
13
14 @Data //getter setter
15 @Entity
16 @NoArgsConstructor
17 @AllArgsConstructor
18
19 public class User {
20     @Id//primary key
21     @GeneratedValue(strategy = GenerationType.IDENTITY)
22     private int userId;
23     private String username;
24     private String password;
25     private String role;//ROLE_USER, ROLE_ADMIN
26
27     @CreationTimestamp
28     private Timestamp createDate;
29 }
```


Prototyping

2 Login Test Case



Controller

```

36 // 시큐리티 설정
37 @GetMapping("/user")
38 public @ResponseBody
39 String user() { return "user"; }
40 // 시큐리티 설정
41 @GetMapping("/admin")
42 public @ResponseBody
43 String admin() { return "admin"; }
44 // 시큐리티 설정
45 @GetMapping("/manager")
46 public @ResponseBody
47 String manager() { return "manager"; }
48 // Spring Security가 해당 주소를 받음
49 @GetMapping("/loginForm")
50 public String loginForm() { return "loginForm"; }
51 // 시큐리티 설정
52 @GetMapping("/joinForm")
53 public String joinForm() { return "joinForm"; }
54 // 시큐리티 설정
55 @PostMapping("/join")
56 public String join(User user) {
57     System.out.println(user);
58     user.setRole("ROLE_USER");
59     String rawPassword = user.getPassword();
60     String encPassword = bCryptPasswordEncoder.encode(rawPassword);
61     user.setPassword(encPassword); // 인코딩 완료해서 setPassword
62     userRepository.save(user); // 회원가입이 잘되지만 시큐리티로는 로그인을 할 수 없다 이유는 패스워드 암호화가 되지 않았기 때문
63     return "redirect:/loginForm"; //redirect:를 붙이면 /loginForm함수를 다시 불러온다.
64 }

```

Prototyping

2 Login Test Case

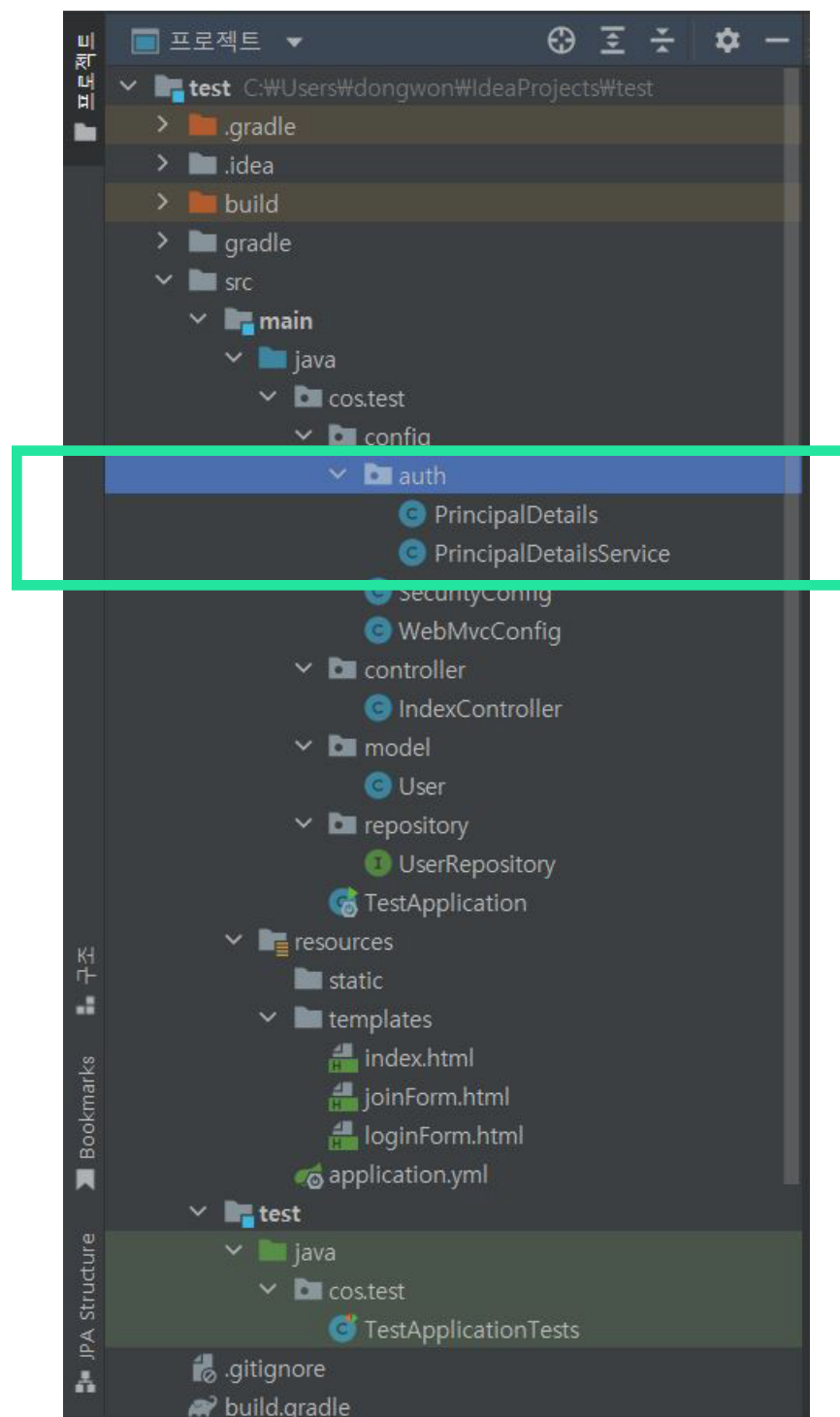
The screenshot shows a web browser window with the title 'JoinForm' and the address bar displaying 'localhost:8080/joinForm'. The page content includes a heading '회원가입 페이지' (Member Registration Page) and a form with two input fields: the first contains 'P123456' and the second contains '....'. Below the form is a button labeled '회원가입' (Join).

Below the form is a 'Result Grid' table with the following data:

	userId	createDate	username	password	role
	1	2022-05-14 15:59:07.903000	B811222	\$2a\$10\$uCkkoE9vapIWldRdZUvXA.4ya01V9W...	ROLE_MANAGER
	2	2022-05-14 16:00:04.957000	C123456	\$2a\$10\$6X1/fFAiLNTouXhJMBRfM.Gxl/atsoUq/...	ROLE_ADMIN
	8	2022-05-21 21:17:32.330000	Q123456	\$2a\$10\$MUjdV0/AXZr8RhIkRz4D0uWUgLOR.2lsu...	ROLE_MANAGER
	9	2022-05-21 21:41:17.416000	P123456	\$2a\$10\$jroQxqmiuinasqczwsnZOdhtUAod/zRk...	ROLE_ADMIN
	10	2022-05-21 22:10:29.955000	P123456	\$2a\$10\$aKvYLrAKmx5YN6ettz0o4ui6cw2jpePuq...	ROLE_USER
	NULL	NULL	NULL	NULL	NULL

Prototyping

2 Login Test Case

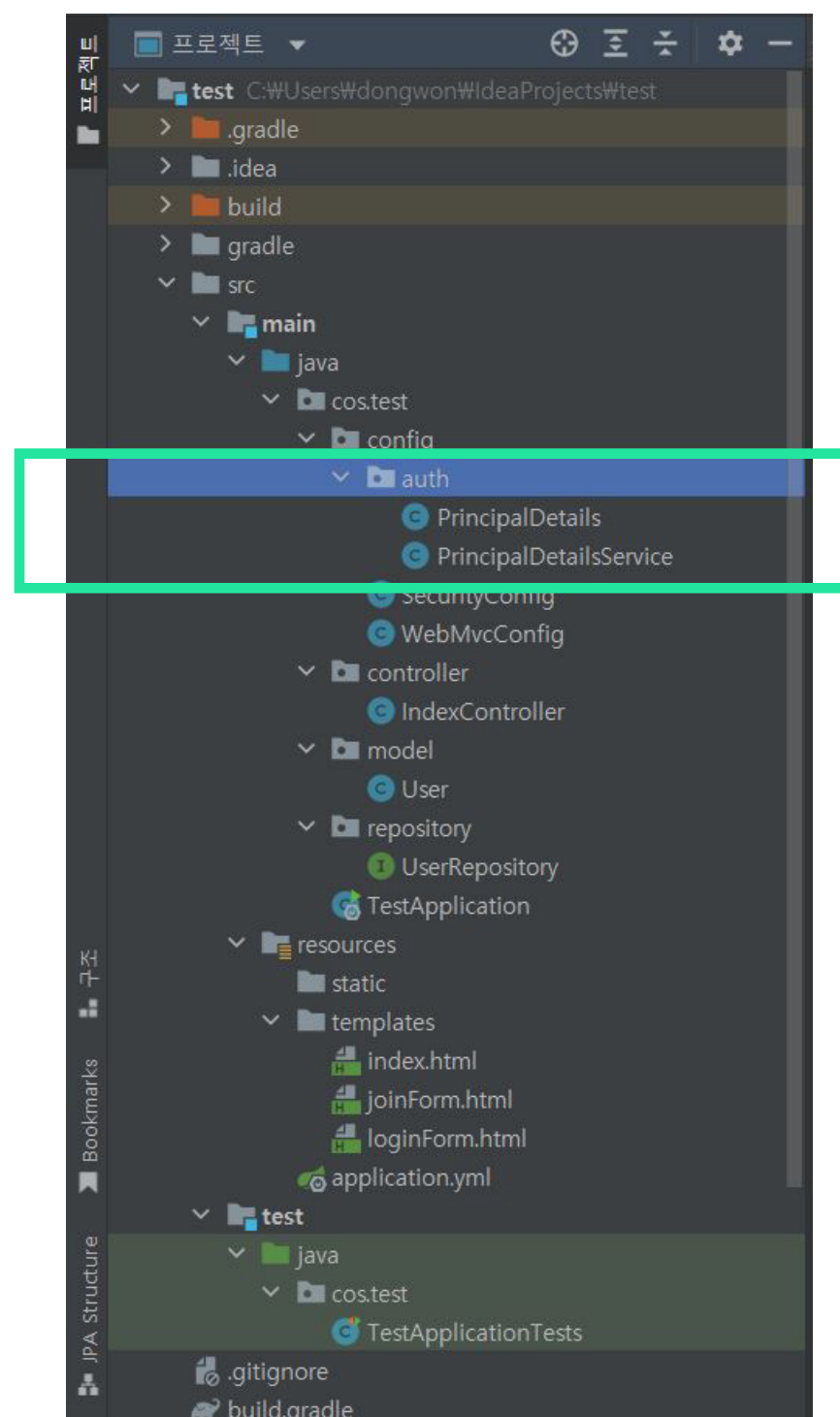


UserDetails

```
18 public class PrincipalDetails implements UserDetails {
19
20     // 우리 유저정보는 User객체가 들고있다.
21     private User user;
22     public PrincipalDetails(User user) { this.user=user; }
23
24     // 해당 유저의 권한을 리턴하는 곳
25     @Override
26     public Collection<? extends GrantedAuthority> getAuthorities() {
27         Collection<GrantedAuthority> collect = new ArrayList<>();
28         collect.add(new GrantedAuthority() {
29             @Override
30             public String getAuthority() { return user.getRole(); }
31         });
32         return collect;
33     }
34
35     @Override
36     public String getPassword() { return user.getPassword(); }
37
38     @Override
39     public String getUsername() { return user.getUsername(); }
40
41     @Override
42     public boolean isAccountNonExpired() { return true; }
43
44     @Override
45     public boolean isAccountNonLocked() { return true; }
46
47     @Override
48     public boolean isCredentialsNonExpired() { return true; }
49
50     @Override
51     public boolean isEnabled() { //계정이 활성화
52         // 우리사이트에서 1년동안 회원이 로그인을 안하면 휴면 계정으로 하기로 함
53         // 현재시간 - 로그인 시간 -> 1년을 초과하면 return false; 이런식으로 구현을 하면 된다.
54         return true;
55     }
56 }
```


Prototyping

2 Login Test Case

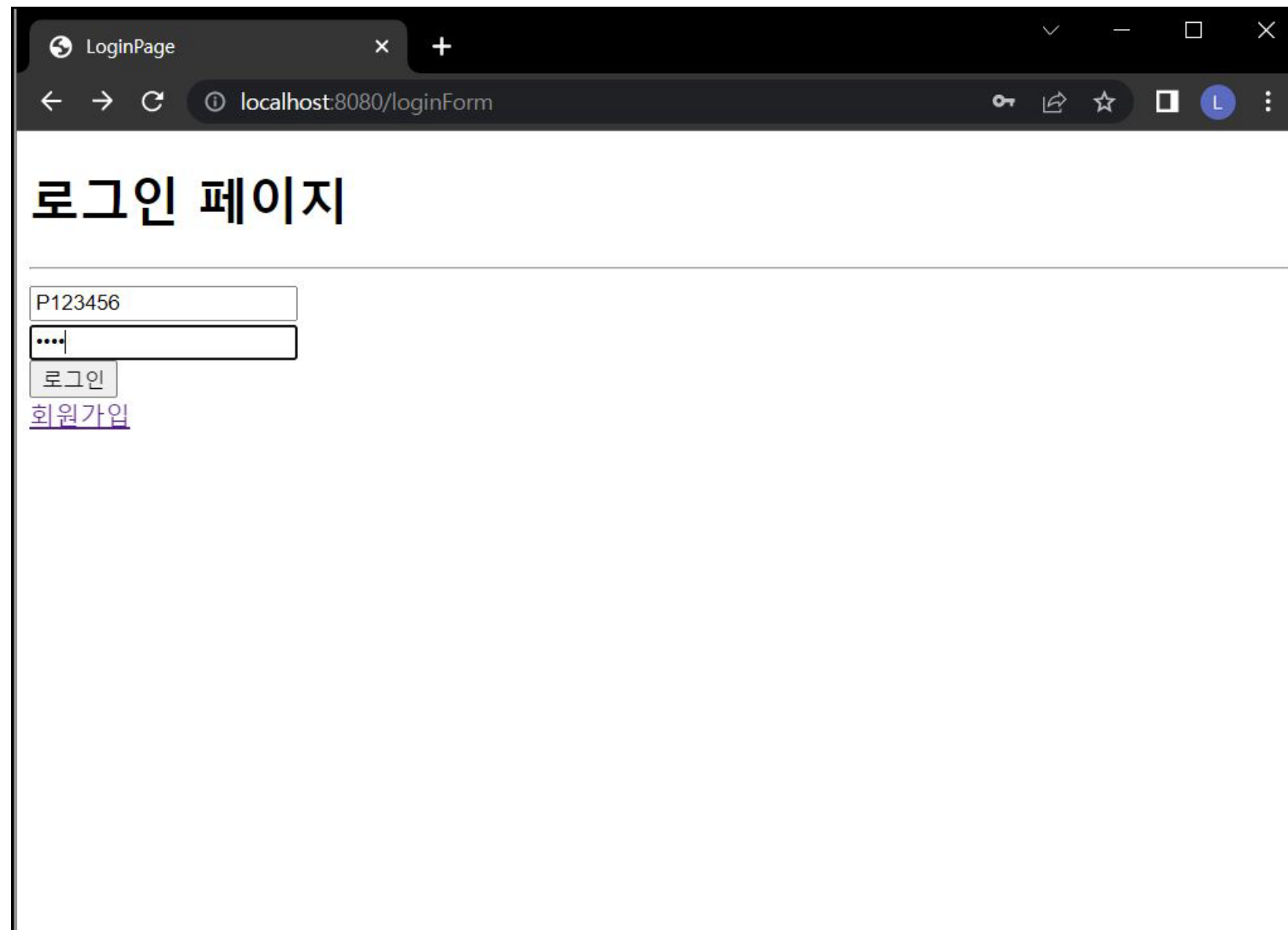


UserDetailsService

```
1 package cos.test.config.auth;
2
3 import cos.test.model.User;
4 import cos.test.repository.UserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.security.core.userdetails.UserDetails;
7 import org.springframework.security.core.userdetails.UserDetailsService;
8 import org.springframework.security.core.userdetails.UsernameNotFoundException;
9 import org.springframework.stereotype.Service;
10
11 // 시큐리티 설정에서 loginProcessUrl("/login")으로 걸어놨기 때문에
12 // /login요청이 오면 자동으로 UserDetailsService 타입으로 IoC되어 있는 loadUserByUsername 함수가 실행
13
14 @Service // PrincipalDetailsService가 IoC에 등록이 된다.
15 public class PrincipalDetailsService implements UserDetailsService {
16
17     @Autowired
18     private UserRepository userRepository;
19     // 시큐리티 세션 -> Authentication -> UserDetails(PrincipalDetails)타입
20
21     @Override
22     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
23         User userEntity = userRepository.findByUsername(username);
24         if (userEntity != null) {
25             return new PrincipalDetails(userEntity);
26         }
27         return null;
28     }
29 }
```


Prototyping

2 Login Test Case



LoginPage

localhost:8080/loginForm

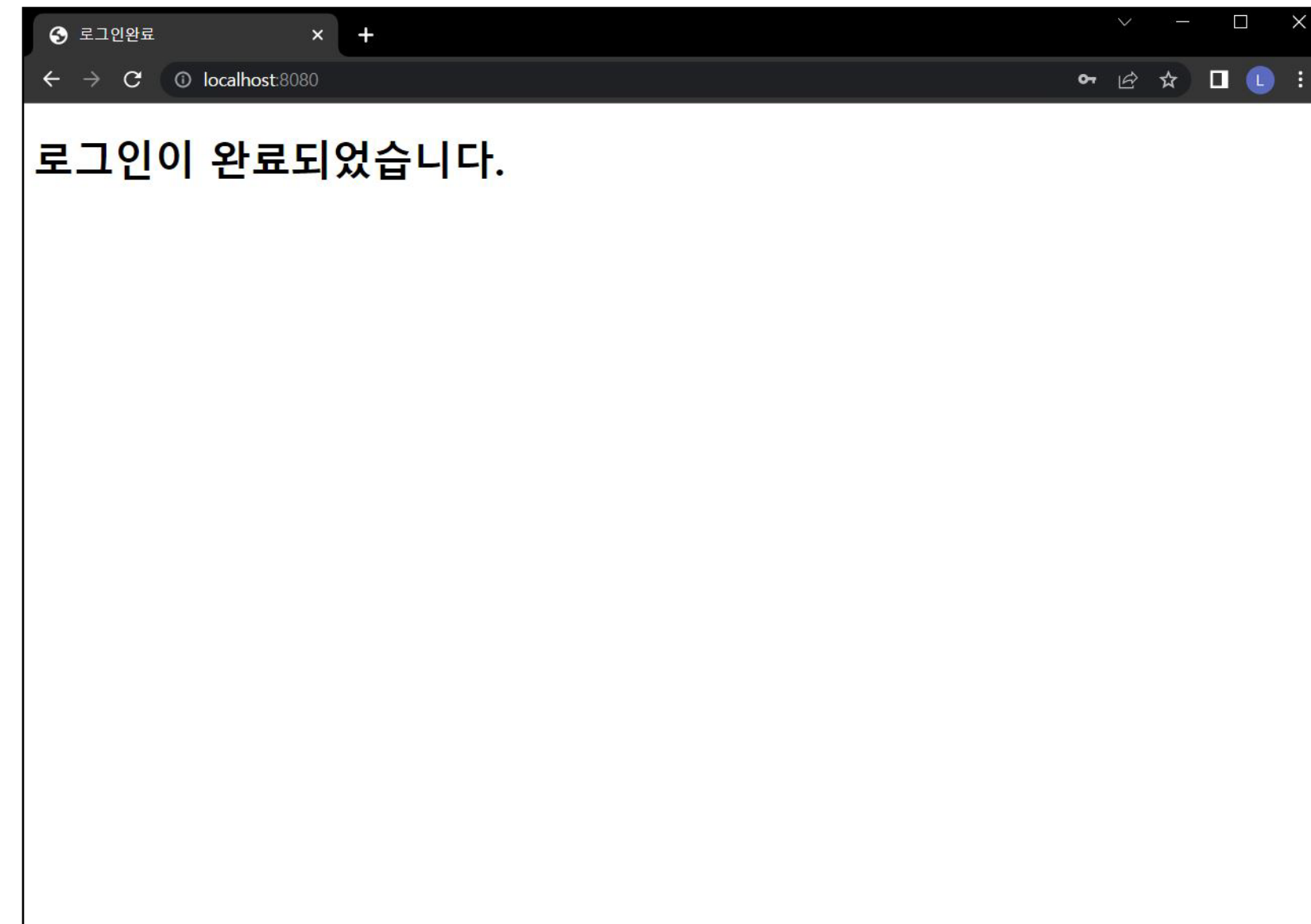
로그인 페이지

P123456

....

로그인

[회원가입](#)



로그인완료

localhost:8080

로그인이 완료되었습니다.

Schedule

MARCH

Ideation

APRIL

Design program architecture and data architecture

MAY

Prototyping page layout
and test case for sign-up
and log-in

JUNE

Developing trial performance

JULY

Page layout design with
HTML, CSS
Algorithm for feedback
event

AUGUST

Develop interaction for rec-
ommendation system
Develop pages for login,
recommendation, timetable,
feedback

SEPTEMBER

Adapting interactions for
user
Develop additional pages

OCTOBER

Efficiency improvements