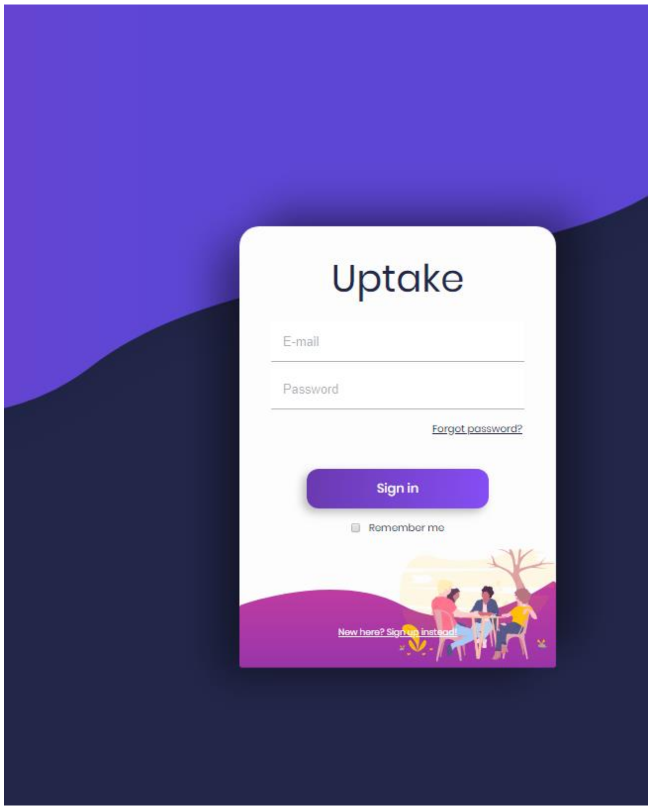


UPTAKE

Social Networking Web App

ABSTRACT
This is the engineering proof of concept. The goal is to exercise the technologies we learnt in this course.

Davaabayar Battogtokkh
CS544 Enterprise Architecture

Contents

Profject Requirement Check List

SN	TASK	PROGRESS
1	Requirement Analyses and create user case	DONE
2	Database design	DONE
3	UI design & navigation	DONE
4	Login & Registration Functionality	DONE
5	Add photo & text post	DONE
6	Add advertisement	DONE
7	Follow user	DONE
8	Notifucation	DONE
9	Like button	DONE
10	Comment	70%
11	Update profile	50%
12	Delete user post	DONE
13	Admin functionality	DONE
14	Search user	DONE
15	Search post	DONE
16	User profile	DONE
17	User home timeline	DONE
18	Show advertisement on users nesfeed	0%
	Validation	DONE
	Features with extra points	
19	Add video and text post	DONE
20	Internalization	DONE

Report Summary or Essay

As a working as a team, the first challenge was to analyse the project scope and devide to task and assign team members. For that we used Use Case diagram as out planning and communitaion tool. Project period was short. Till mid term we learnt fundamentals of Spring technology and each member worked on their labs to understand and concepts, like: DI, AOP, ORM, JPA, Spring MVC, Thymeleaf, Spring boot, Spring Data, Spring Security, Messaging, JWT.

From the 3rd week we started on working on the project. As we had exact 7 days to work on the project, we needed effective SDLC. Using our knowledge gained from SE course and previous work experience we used our version of Scrum Tool which you call it **“Use case based Task Planning / Tracking”.**

Each day we meet or chat and discuss previous days progress and assign a task, and made required changes to the use case. This method worked very well for us, because it helped us to see the project scope as a whole and monitor the project progress.

“Use case based Task Planning / Tracking diagrams” are included, as it was interesting for us to see how it expands.

Analysis phase: Use case level 1

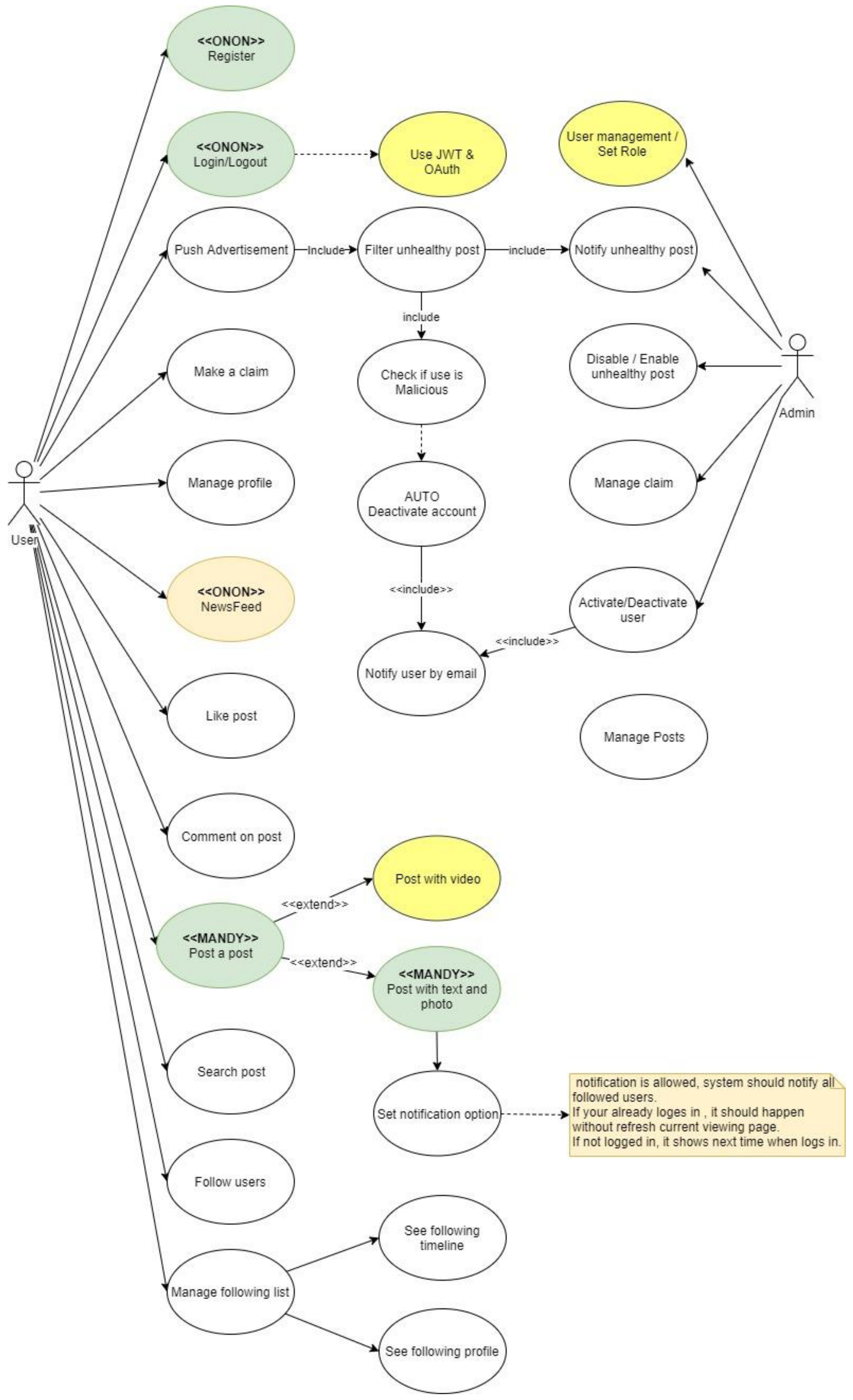
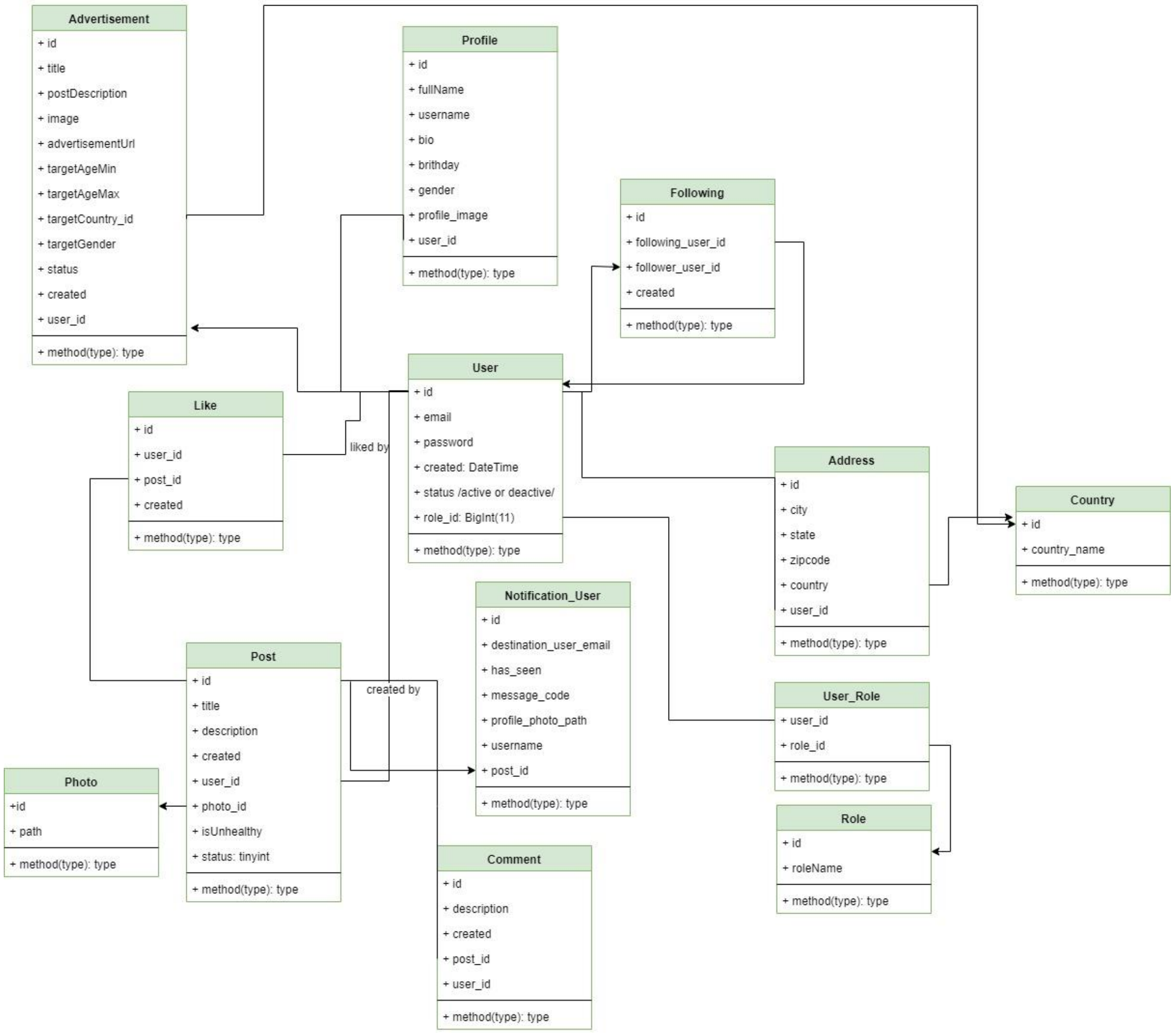


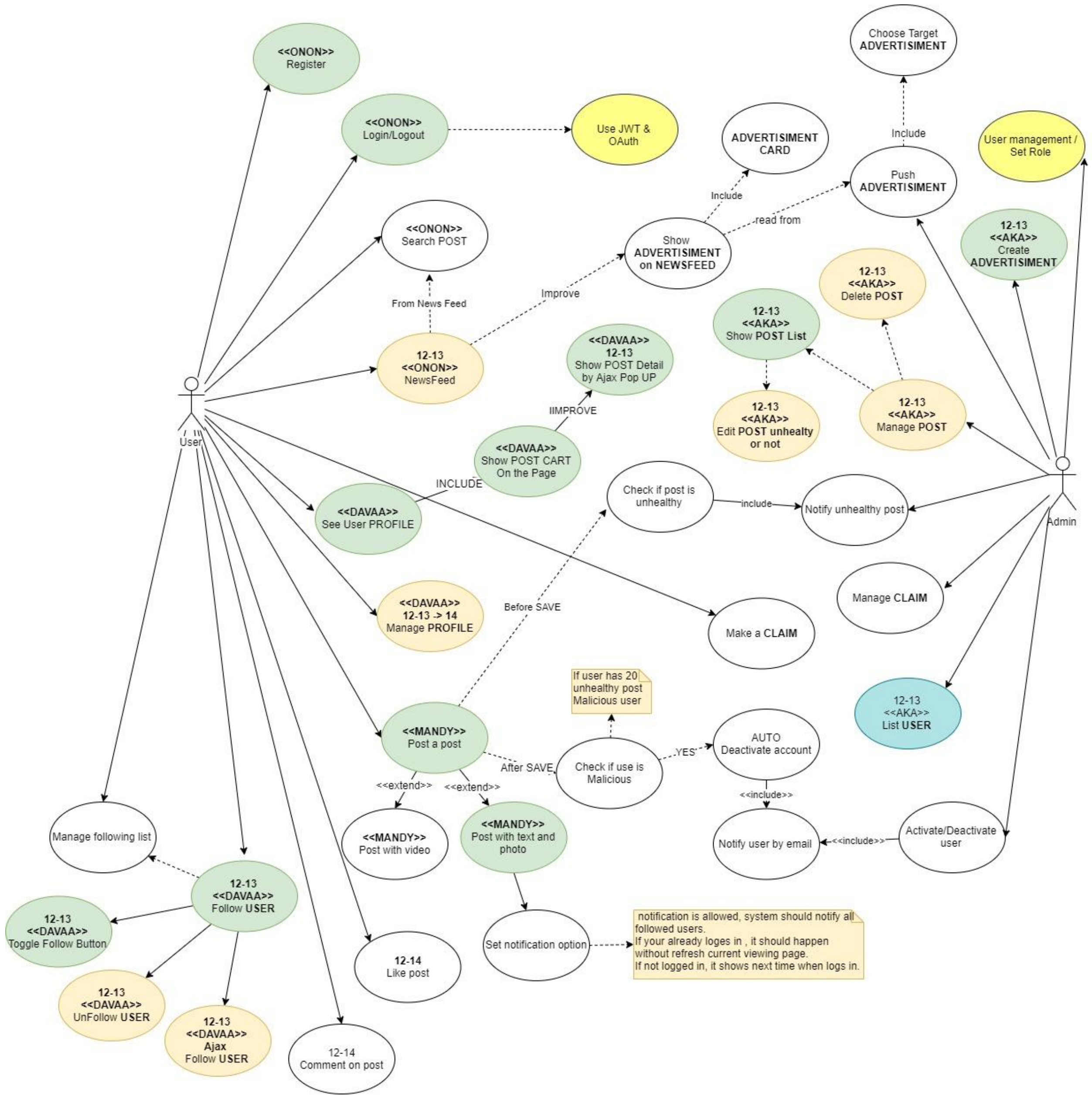
Figure 1Use Case diagram Level 1

Domain object model or Entity Relatioin diagram



Use case based Task Planning / Tracking diagram

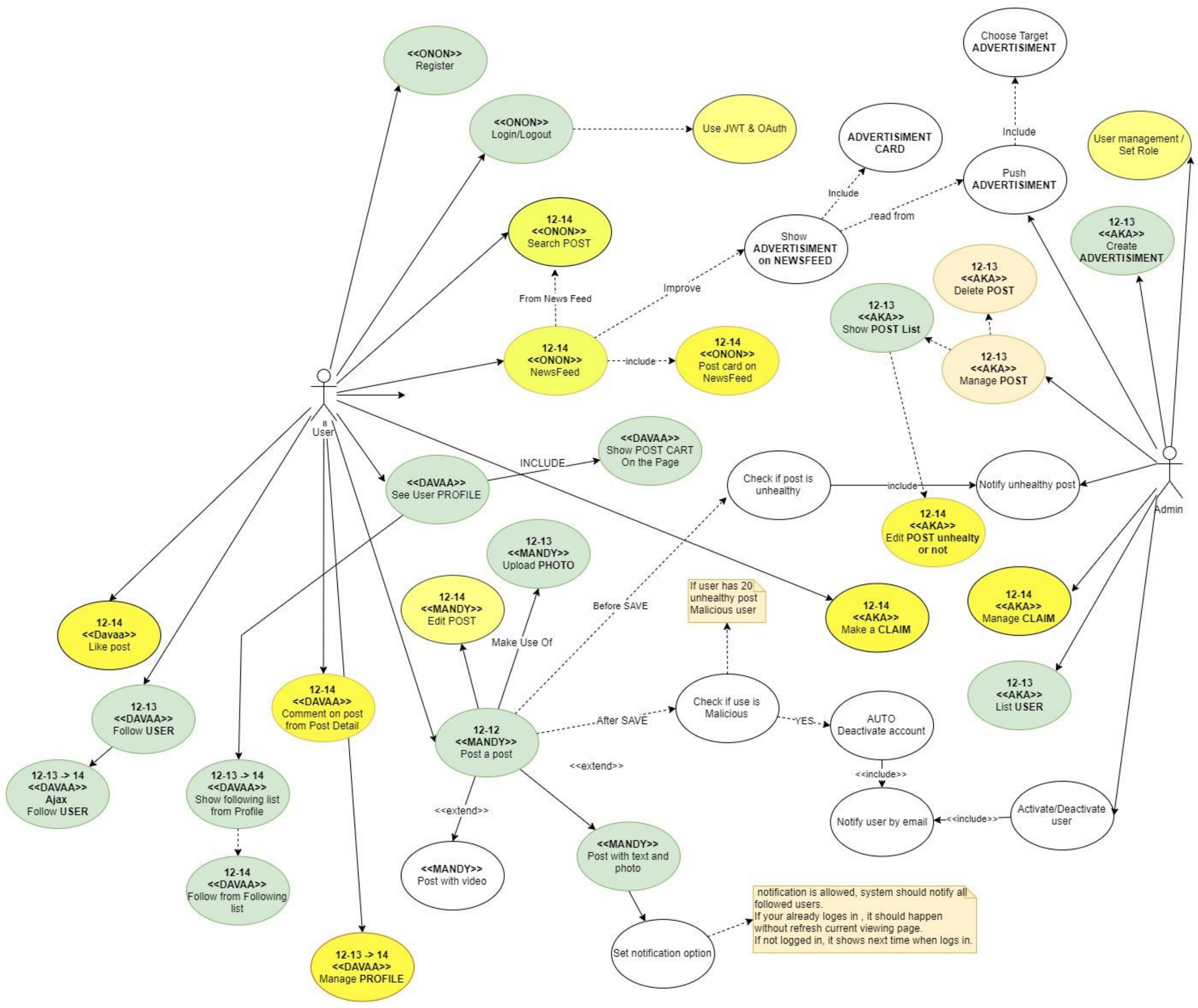
Friday 13



<<ONON>> Register
<<ONON>> Login/Logout
Use JWT & OAuth
<<ONON>> Search POST
From News Feed
12-13 <<ONON>> NewsFeed
Improve
Show ADVERTISIMENT on NEWSFEED
Include
ADVERTISIMENT CARD
read from
Push ADVERTISIMENT
Include
Choose Target ADVERTISIMENT
User management / Set Role
12-13 <<AKA>> Create ADVERTISIMENT
12-13 <<AKA>> Delete POST
12-13 <<AKA>> Show POST List
12-13 <<AKA>> Edit POST unhealty or not
12-13 <<AKA>> Manage POST
<<DAVAA>> 12-13 Show POST Detail by Ajax Pop UP
IIMPROVE
<<DAVAA>> Show POST CART On the Page
INCLUDE
<<DAVAA>> See User PROFILE
User
Admin
Check if post is unhealthy
include
Notify unhealthy post
Manage CLAIM
Make a CLAIM
Before SAVE
<<DAVAA>> 12-13 -> 14 Manage PROFILE
12-13 <<AKA>> List USER
If user has 20 unhealthy post Malicious user
<<MANDY>> Post a post
After SAVE
Check if use is Malicious
YES
AUTO Deactivate account
<<include>>
Notify user by email
<<include>>
Activate/Deactivate user
<<extend>>
<<extend>>
<<MANDY>> Post with video
<<MANDY>> Post with text and photo
Set notification option
notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.
Manage following list
12-13 <<DAVAA>> Follow USER
12-13 <<DAVAA>> Toggle Follow Button
12-13 <<DAVAA>> UnFollow USER
12-13 <<DAVAA>> Ajax Follow USER
12-14 Comment on post
12-14 Like post

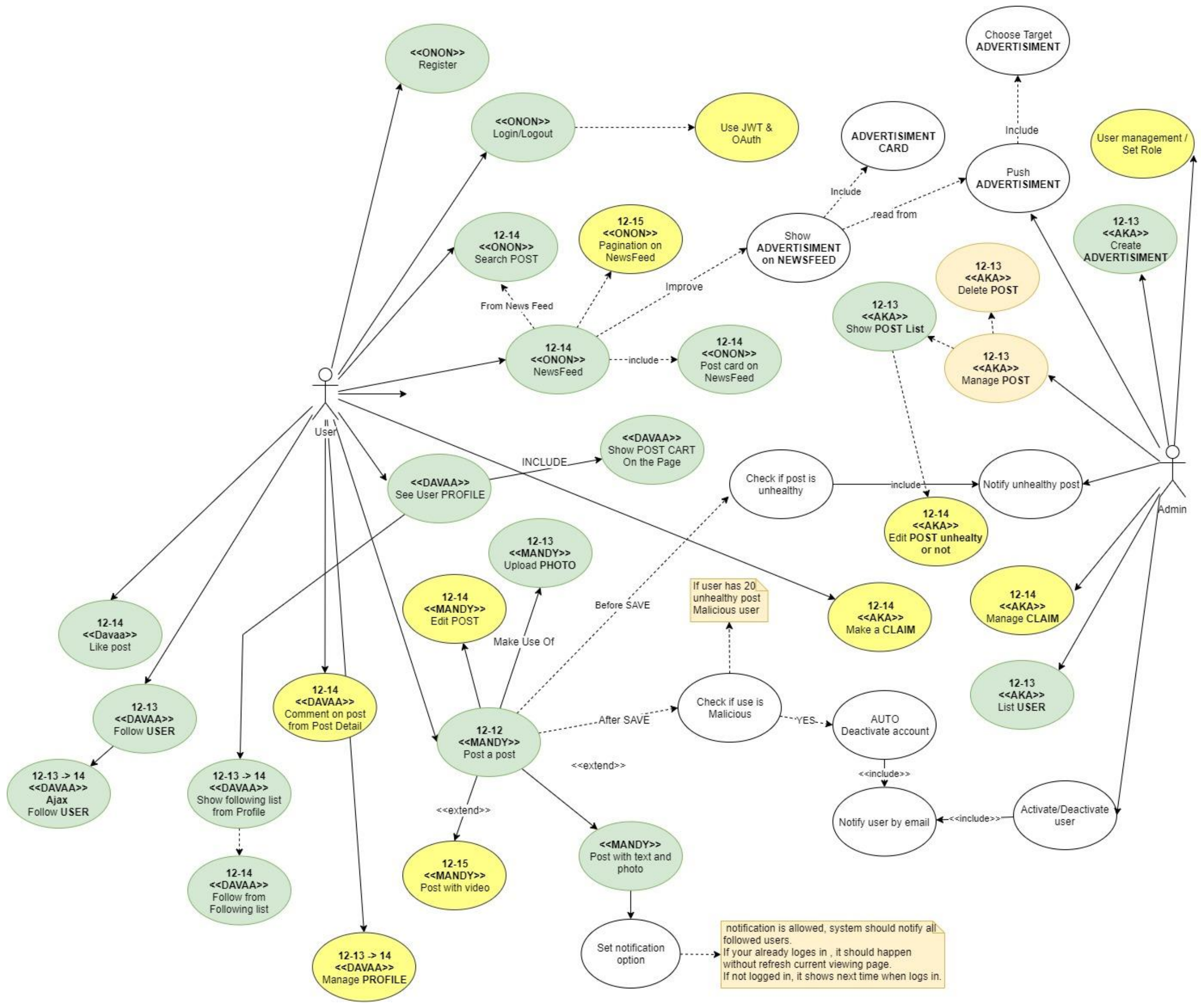
Saturday 14. Initial Use case based Task Planning and Tracking diagram.

Initial use cases are divided into tasks and planned.



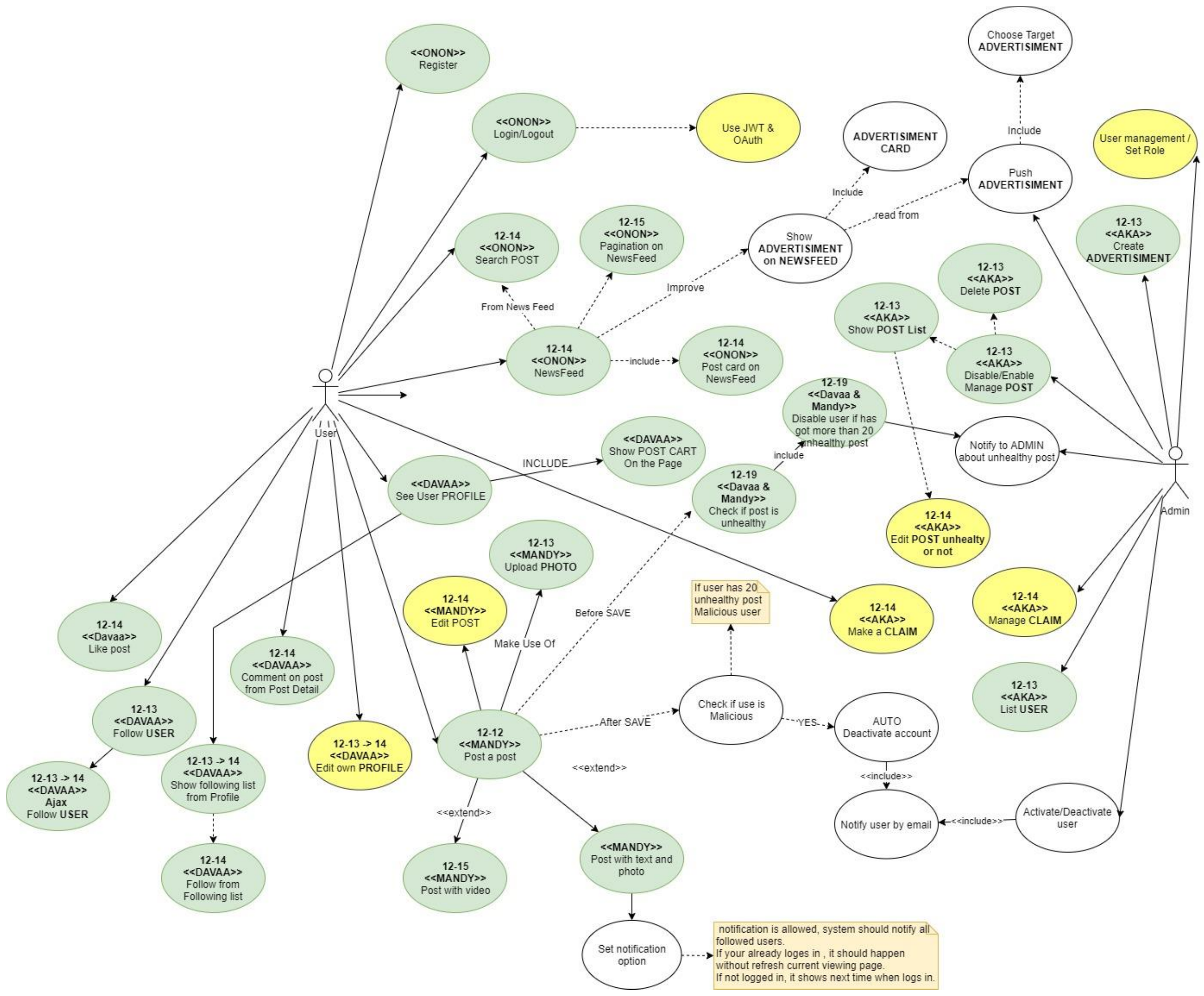
<<ONON>> Register
<<ONON>> Login/Logout
Use JWT & OAuth
12-14 <<ONON>> Search POST
From News Feed
12-14 <<ONON>> NewsFeed
include
12-14 <<ONON>> Post card on NewsFeed
Improve
Show ADVERTISIMENT on NEWSFEED
Include
ADVERTISIMENT CARD
read from
Push ADVERTISIMENT
Include
Choose Target ADVERTISIMENT
User management / Set Role
12-13 <<AKA>> Create ADVERTISIMENT
12-13 <<AKA>> Delete POST
12-13 <<AKA>> Show POST List
12-13 <<AKA>> Manage POST
User
Admin
<<DAVAA>> See User PROFILE
INCLUDE
<<DAVAA>> Show POST CART On the Page
Check if post is unhealthy
include
Notify unhealthy post
12-14 <<AKA>> Edit POST unhealty or not
12-14 <<AKA>> Manage CLAIM
12-14 <<AKA>> Make a CLAIM
12-13 <<AKA>> List USER
12-13 <<MANDY>> Upload PHOTO
12-14 <<MANDY>> Edit POST
Make Use Of
Before SAVE
If user has 20 unhealthy post Malicious user
12-14 <<Davaa>> Like post
12-13 <<DAVAA>> Follow USER
12-13 -> 14 <<DAVAA>> Ajax Follow USER
12-14 <<DAVAA>> Comment on post from Post Detail
12-13 -> 14 <<DAVAA>> Show following list from Profile
12-14 <<DAVAA>> Follow from Following list
12-13 -> 14 <<DAVAA>> Manage PROFILE
12-12 <<MANDY>> Post a post
After SAVE
Check if use is Malicious
YES
AUTO Deactivate account
<<include>>
Notify user by email
<<include>>
Activate/Deactivate user
<<extend>>
<<extend>>
<<MANDY>> Post with video
<<MANDY>> Post with text and photo
Set notification option
notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.

Sunday 15



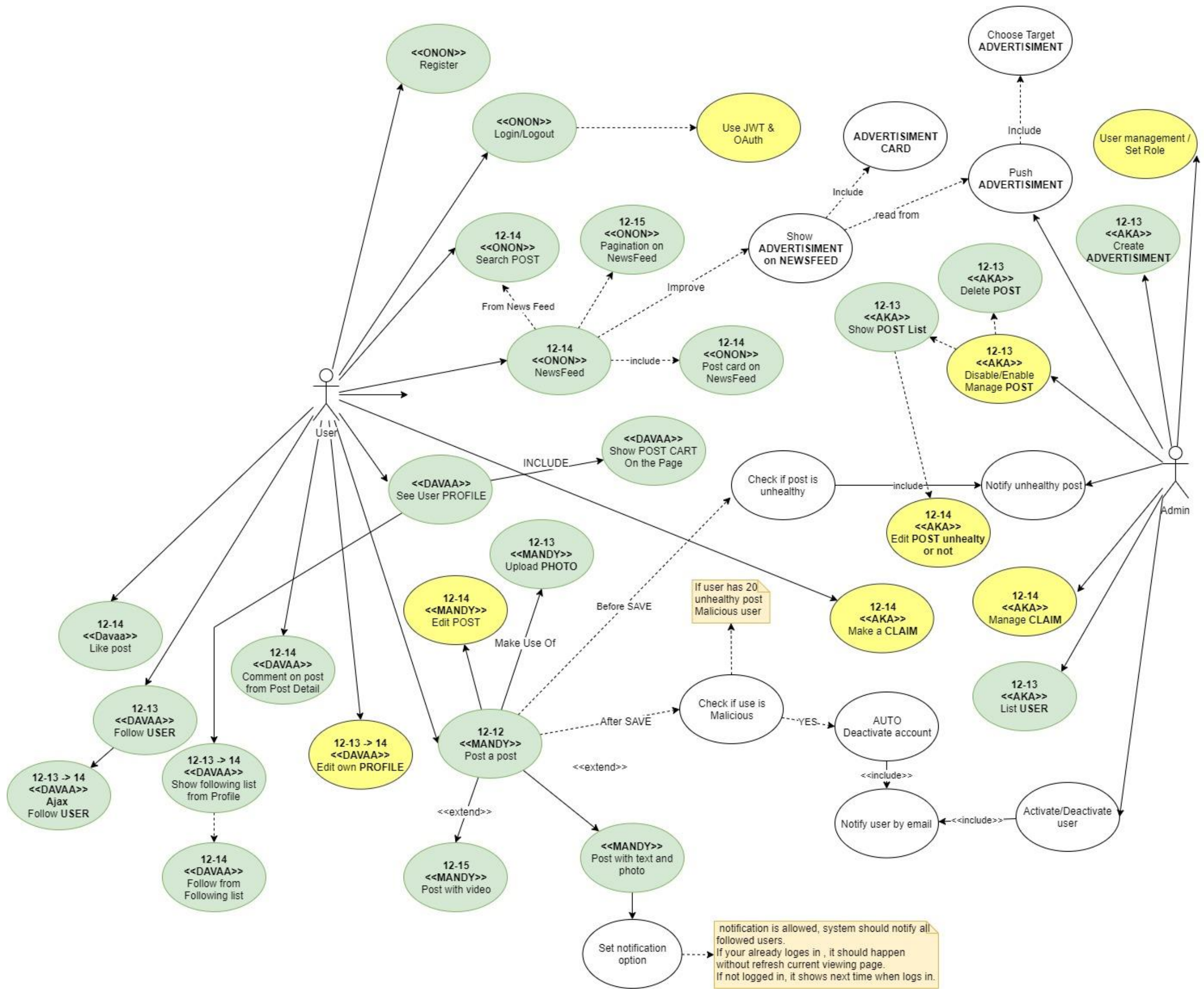
<<ONON>> Register
<<ONON>> Login/Logout
Use JWT & OAuth
12-14 <<ONON>> Search POST
12-15 <<ONON>> Pagination on NewsFeed
Improve
From News Feed
12-14 <<ONON>> NewsFeed
include
12-14 <<ONON>> Post card on NewsFeed
Show ADVERTISIMENT on NEWSFEED
Include
ADVERTISIMENT CARD
read from
Push ADVERTISIMENT
Include
Choose Target ADVERTISIMENT
User management / Set Role
12-13 <<AKA>> Create ADVERTISIMENT
12-13 <<AKA>> Delete POST
12-13 <<AKA>> Show POST List
12-13 <<AKA>> Manage POST
User
Admin
INCLUDE
<<DAVAA>> Show POST CART On the Page
<<DAVAA>> See User PROFILE
Check if post is unhealthy
include
Notify unhealthy post
12-14 <<AKA>> Edit POST unhealty or not
12-13 <<MANDY>> Upload PHOTO
12-14 <<MANDY>> Edit POST
Make Use Of
Before SAVE
If user has 20 unhealthy post Malicious user
12-14 <<AKA>> Make a CLAIM
12-14 <<AKA>> Manage CLAIM
12-14 <<Davaa>> Like post
12-13 <<DAVAA>> Follow USER
12-14 <<DAVAA>> Comment on post from Post Detail
12-12 <<MANDY>> Post a post
After SAVE
Check if use is Malicious
YES
AUTO Deactivate account
12-13 <<AKA>> List USER
<<extend>>
12-13 -> 14 <<DAVAA>> Ajax Follow USER
12-13 -> 14 <<DAVAA>> Show following list from Profile
<<include>>
Notify user by email
<<include>>
Activate/Deactivate user
<<extend>>
12-14 <<DAVAA>> Follow from Following list
12-15 <<MANDY>> Post with video
<<MANDY>> Post with text and photo
Set notification option
notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.
12-13 -> 14 <<DAVAA>> Manage PROFILE

Monday 16



<<ONON>> Register
<<ONON>> Login/Logout
Use JWT & OAuth
Choose Target ADVERTISIMENT
Include
ADVERTISIMENT CARD
Push ADVERTISIMENT
User management / Set Role
Include
read from
12-14 <<ONON>> Search POST
12-15 <<ONON>> Pagination on NewsFeed
Show ADVERTISIMENT on NEWSFEED
12-13 <<AKA>> Create ADVERTISIMENT
Improve
12-13 <<AKA>> Delete POST
From News Feed
12-13 <<AKA>> Show POST List
12-14 <<ONON>> NewsFeed
include
12-14 <<ONON>> Post card on NewsFeed
12-13 <<AKA>> Disable/Enable Manage POST
12-19 <<Davaa & Mandy>> Disable user if has got more than 20 unhealthy post
User
Notify to ADMIN about unhealthy post
<<DAVAA>> Show POST CART On the Page
INCLUDE
include
<<DAVAA>> See User PROFILE
12-19 <<Davaa & Mandy>> Check if post is unhealthy
Admin
12-14 <<AKA>> Edit POST unhealty or not
12-13 <<MANDY>> Upload PHOTO
12-14 <<MANDY>> Edit POST
Before SAVE
If user has 20 unhealthy post Malicious user
12-14 <<AKA>> Make a CLAIM
12-14 <<AKA>> Manage CLAIM
12-14 <<Davaa>> Like post
Make Use Of
12-14 <<DAVAA>> Comment on post from Post Detail
12-13 <<AKA>> List USER
12-13 <<DAVAA>> Follow USER
Check if use is Malicious
12-13 -> 14 <<DAVAA>> Edit own PROFILE
After SAVE
12-12 <<MANDY>> Post a post
YES
AUTO Deactivate account
12-13 -> 14 <<DAVAA>> Ajax Follow USER
12-13 -> 14 <<DAVAA>> Show following list from Profile
<<extend>>
<<include>>
<<extend>>
Notify user by email
<<include>>
Activate/Deactivate user
12-14 <<DAVAA>> Follow from Following list
12-15 <<MANDY>> Post with video
<<MANDY>> Post with text and photo
Set notification option
notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.

Tuesday 17

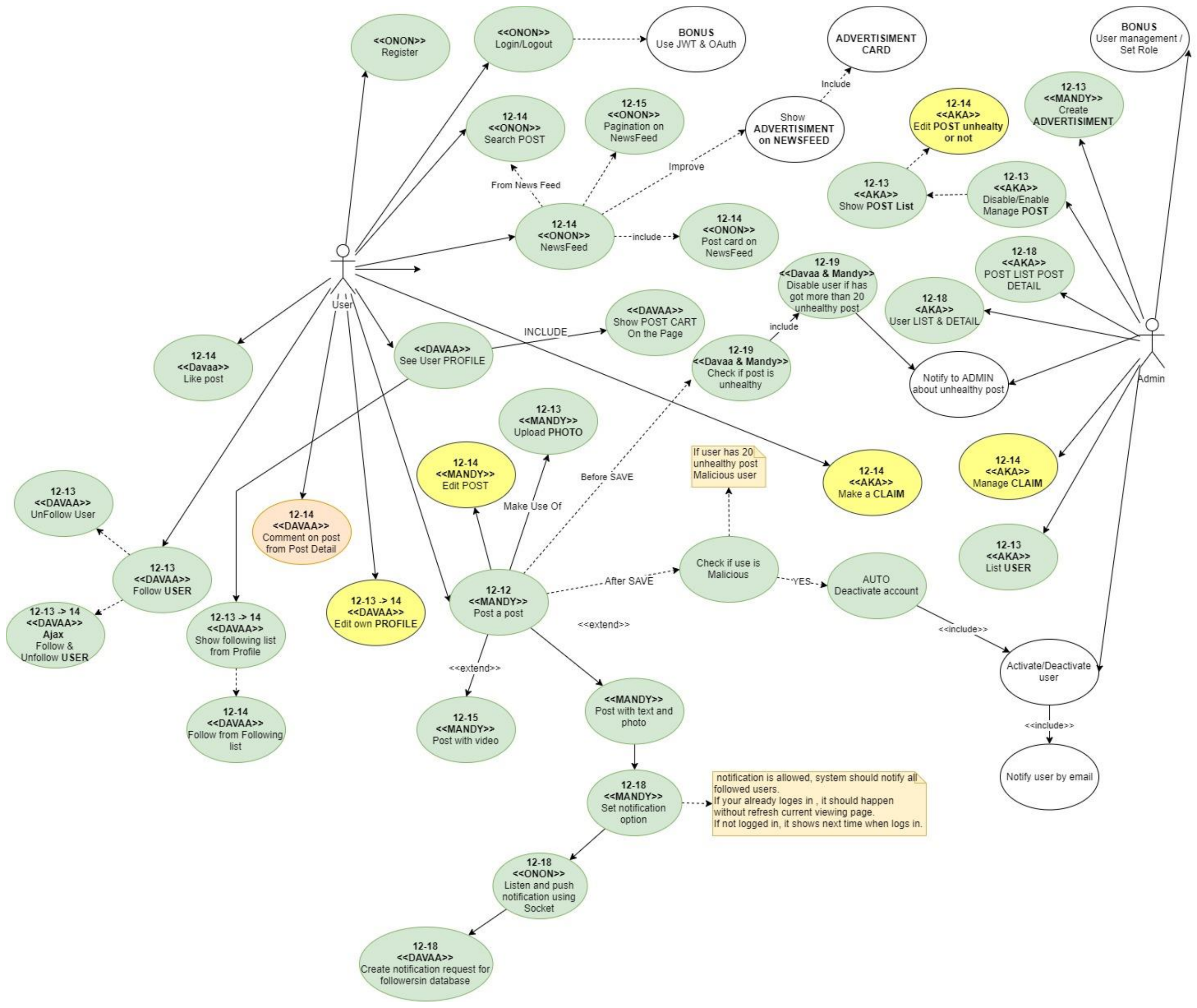


<<ONON>> Register
<<ONON>> Login/Logout
Use JWT & OAuth
ADVERTISIMENT CARD
Choose Target ADVERTISIMENT
Include
Push ADVERTISIMENT
User management / Set Role
Include
read from
Show ADVERTISIMENT on NEWSFEED
12-14 <<ONON>> Search POST
12-15 <<ONON>> Pagination on NewsFeed
Improve
12-13 <<AKA>> Create ADVERTISIMENT
12-13 <<AKA>> Delete POST
12-13 <<AKA>> Show POST List
From News Feed
12-14 <<ONON>> NewsFeed
include
12-14 <<ONON>> Post card on NewsFeed
12-13 <<AKA>> Disable/Enable Manage POST
User
<<DAVAA>> Show POST CART On the Page
INCLUDE
<<DAVAA>> See User PROFILE
Check if post is unhealthy
include
Notify unhealthy post
Admin
12-14 <<AKA>> Edit POST unhealty or not
12-13 <<MANDY>> Upload PHOTO
12-14 <<MANDY>> Edit POST
Before SAVE
12-14 <<AKA>> Make a CLAIM
12-14 <<AKA>> Manage CLAIM
12-14 <<Davaa>> Like post
Make Use Of
12-14 <<DAVAA>> Comment on post from Post Detail
12-13 <<DAVAA>> Follow USER
12-13 <<AKA>> List USER
Check if use is Malicious
YES
AUTO Deactivate account
After SAVE
12-13 -> 14 <<DAVAA>> Edit own PROFILE
12-12 <<MANDY>> Post a post
12-13 -> 14 <<DAVAA>> Show following list from Profile
12-13 -> 14 <<DAVAA>> Ajax Follow USER
<<extend>>
<<include>>
<<extend>>
Notify user by email
<<include>>
Activate/Deactivate user
12-14 <<DAVAA>> Follow from Following list
12-15 <<MANDY>> Post with video
<<MANDY>> Post with text and photo
Set notification option

If user has 20 unhealthy post Malicious user

notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.

Wednesday 18 – Thursday 19 | Final result



<<ONON>> Register
<<ONON>> Login/Logout
BONUS Use JWT & OAuth
ADVERTISIMENT CARD
BONUS User management / Set Role
12-15 <<ONON>> Pagination on NewsFeed
12-14 <<ONON>> Search POST
Include
Show ADVERTISIMENT on NEWSFEED
12-14 <<AKA>> Edit POST unhealty or not
12-13 <<MANDY>> Create ADVERTISIMENT
Improve
From News Feed
12-13 <<AKA>> Show POST List
12-13 <<AKA>> Disable/Enable Manage POST
12-14 <<ONON>> NewsFeed
include
12-14 <<ONON>> Post card on NewsFeed
12-19 <<Davaa & Mandy>> Disable user if has got more than 20 unhealthy post
12-18 <<AKA>> POST LIST POST DETAIL
12-18 <AKA>> User LIST & DETAIL
User
INCLUDE
<<DAVAA>> Show POST CART On the Page
include
12-14 <<Davaa>> Like post
<<DAVAA>> See User PROFILE
12-19 <<Davaa & Mandy>> Check if post is unhealthy
Notify to ADMIN about unhealthy post
Admin
12-13 <<MANDY>> Upload PHOTO
If user has 20 unhealthy post Malicious user
12-14 <<MANDY>> Edit POST
Before SAVE
Make Use Of
12-14 <<AKA>> Make a CLAIM
12-14 <<AKA>> Manage CLAIM
12-13 <<DAVAA>> UnFollow User
12-14 <<DAVAA>> Comment on post from Post Detail
12-13 <<DAVAA>> Follow USER
Check if use is Malicious
After SAVE
YES
AUTO Deactivate account
12-13 <<AKA>> List USER
12-13 -> 14 <<DAVAA>> Ajax Follow & Unfollow USER
12-13 -> 14 <<DAVAA>> Show following list from Profile
12-13 -> 14 <<DAVAA>> Edit own PROFILE
12-12 <<MANDY>> Post a post
<<extend>>
<<include>>
Activate/Deactivate user
<<extend>>
<<MANDY>> Post with text and photo
12-15 <<MANDY>> Post with video
12-14 <<DAVAA>> Follow from Following list
<<include>>
Notify user by email
12-18 <<MANDY>> Set notification option
notification is allowed, system should notify all followed users.
If your already loges in , it should happen without refresh current viewing page.
If not logged in, it shows next time when logs in.
12-18 <<ONON>> Listen and push notification using Socket
12-18 <<DAVAA>> Create notification request for followersin database

My Tasks

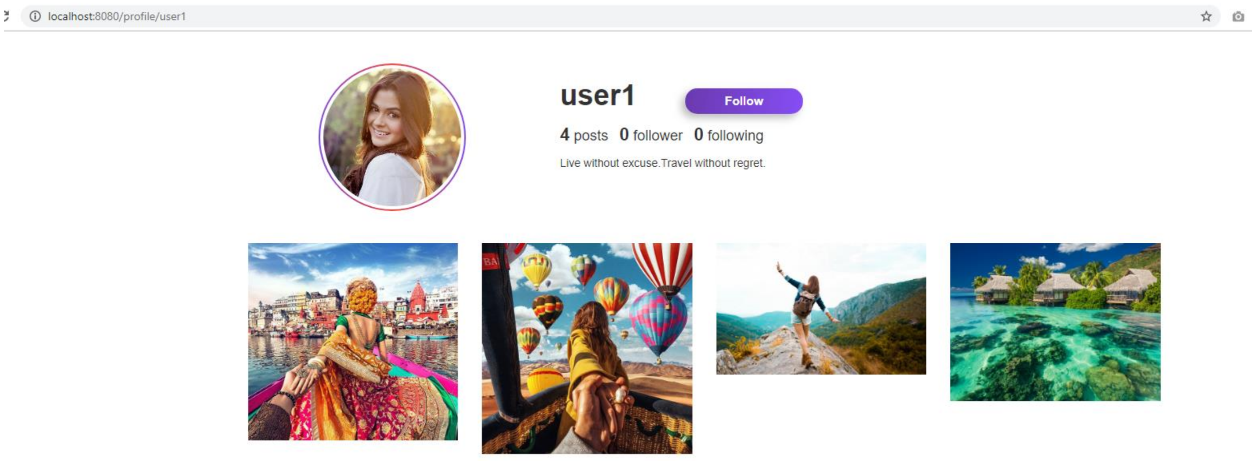
My task completion percentage

SN	TASK	PROGRESS
1	Requirement Analyses and create user case	100%
2	Database design	100%
3	UI design & navigation	100%
4	Manage profile	30%
5	Follow and Unfollow user	100%
6	Manage following list	100%
7	Record Notifucation record when notify to follower option is checked	100%
8	Like button	100%
9	Comment	70%
10	User profile	100%
11	Internalization	100%

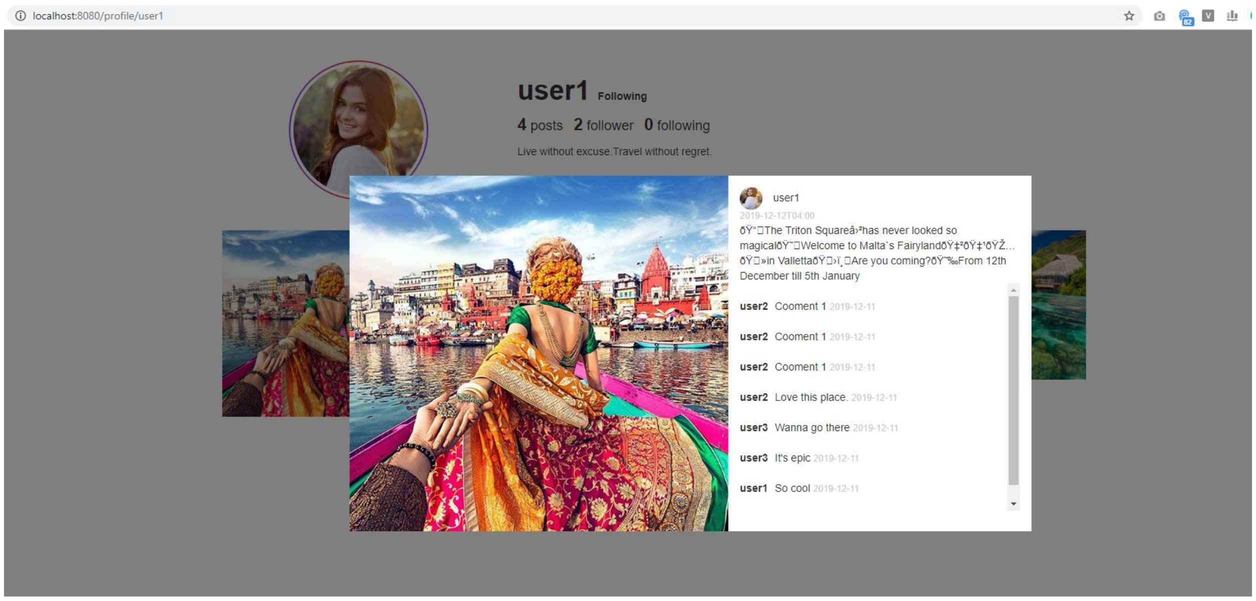
Screen shots

Some of the screenshots I got during the development.

12-13 User profile page


localhost:8080/profile/user1
user1
Follow
4 posts 0 follower 0 following
Live without excuse.Travel without regret.

12-13 Pop up post card



localhost:8080/profile/user1

user1 Following

4 posts 2 follower 0 following

Live without excuse.Travel without regret.

user1

2019-12-12T04:00

ðŸ"The Triton Squareâ›²has never looked so magicalðŸ˜Welcome to Malta`s FairylandðŸ‡²ðŸ‡¹ðŸŽ… ðŸ»in VallettaðŸ›ï¸Are you coming?ðŸ˜‰From 12th December till 5th January

user2 Cooment 1 2019-12-11

user2 Cooment 1 2019-12-11

user2 Cooment 1 2019-12-11

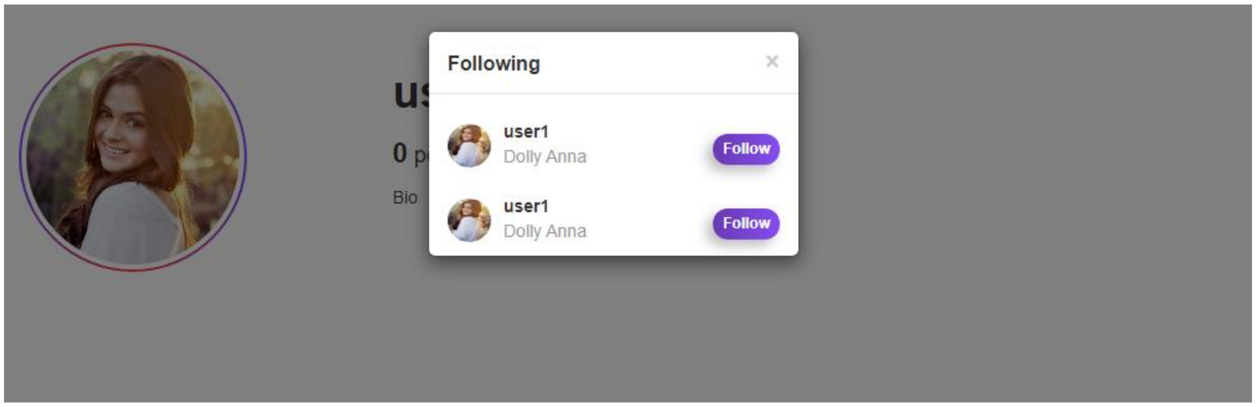
user2 Love this place. 2019-12-11

user3 Wanna go there 2019-12-11

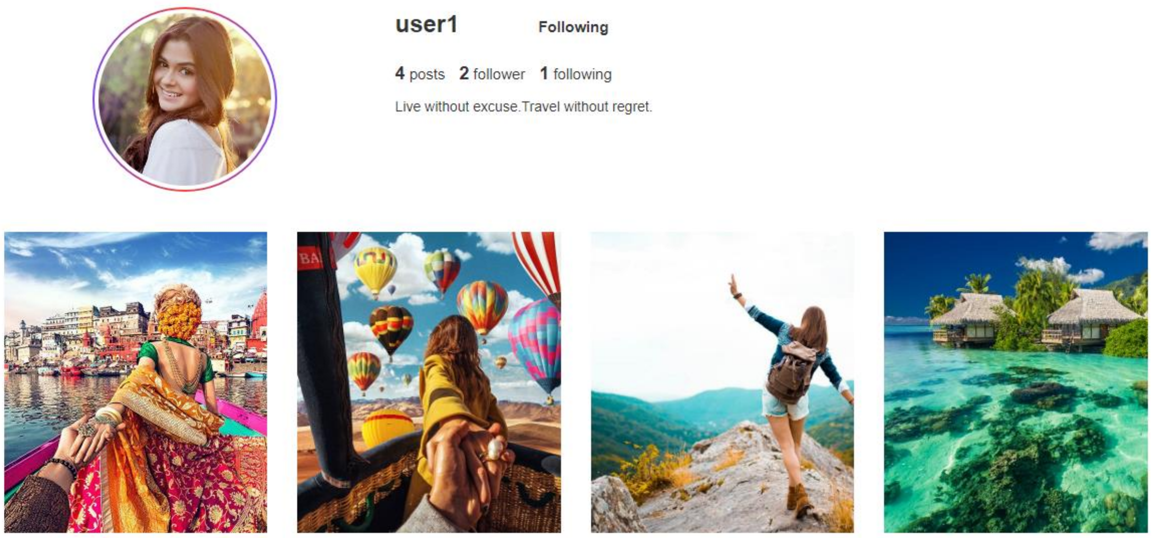
user3 It's epic 2019-12-11

user1 So cool 2019-12-11

12-13 Following ajax list from profile list



12-15 Follow from Following pop list

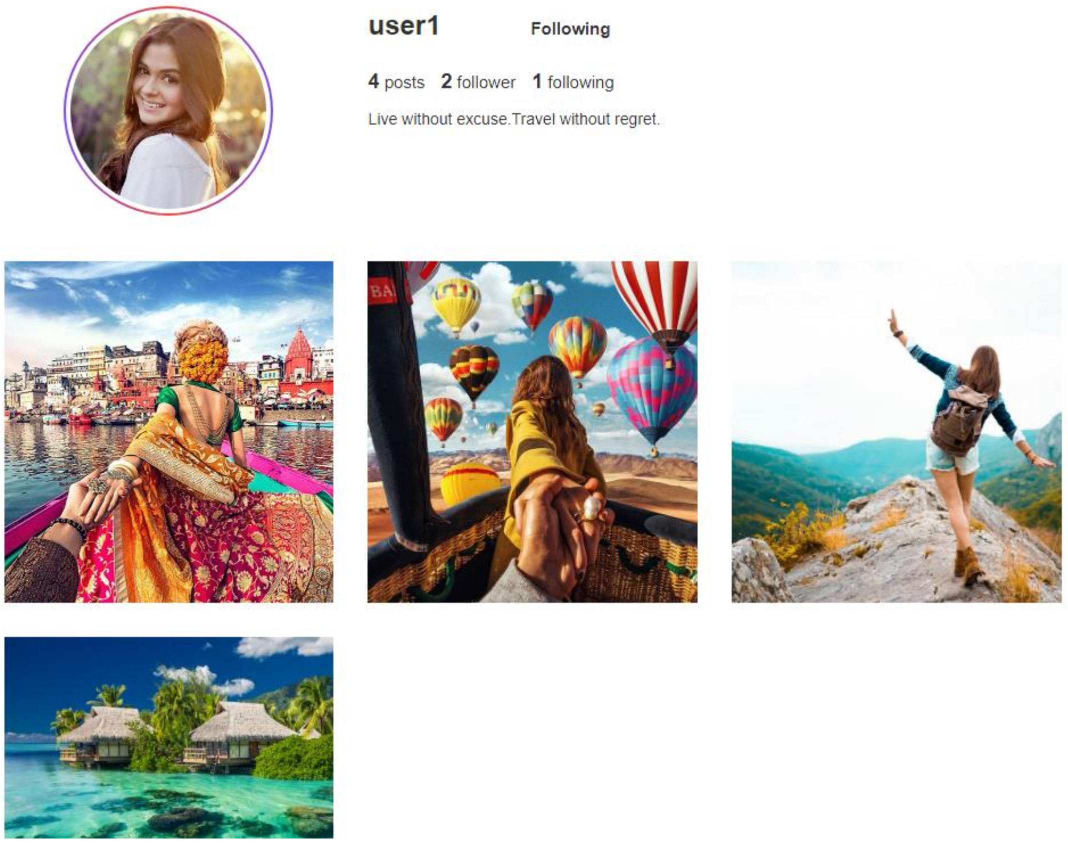


user1 Following

4 posts 2 follower 1 following

Live without excuse.Travel without regret.

12-15 Profile page update



user1 Following

4 posts 2 follower 1 following

Live without excuse.Travel without regret.

Coding

Follow functionality

User Entity

```
@Entity
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "username", unique = true)
    private String username;

    @Email
    @NotBlank
    @Column(name = "email", unique = true)
    private String email;

    @NotBlank
    @Size(min = 6)
    private String password;

    @Transient
    private String passwordConfirm;

    @CreatedDate
    private LocalDate createdDate;

    private boolean status;

    private String publicName;

    @OneToOne(cascade = CascadeType.ALL)
    private Address address;

    @JsonIgnore
    @OneToMany(mappedBy = "followedUser")
    @LazyCollection(LazyCollectionOption.EXTRA)
    private List<Follower> followedUsers;

    @JsonIgnore
    @OneToMany(mappedBy = "followingUser")
    @LazyCollection(LazyCollectionOption.EXTRA)
    private List<Follower> followingUser;

    @ManyToMany
    @JoinTable(name = "user_role", joinColumns = {
            @JoinColumn(name = "user_id", referencedColumnName = "id")},
inverseJoinColumns = {
            @JoinColumn(name = "role_id", referencedColumnName = "id")})
    @JsonIgnore
    private List<Role> role;

    @JsonIgnore
    @OneToMany(mappedBy = "user")
    @LazyCollection(value = LazyCollectionOption.EXTRA)
    private List<Post> postList;

    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private Profile profile;


….

}
```

Follower

```
@Entity
@Table(name = "follower")
public class Follower {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="following_user_id")
    private User followingUser;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="followed_user_id")
    private User followedUser;

    public Follower() {
    }

    public Follower(User followingUser, User followedUser) {
        this.followingUser = followingUser;
        this.followedUser = followedUser;
    }
```

Follow Repository

```
@Repository
public interface FollowRepository extends JpaRepository<Follower, Long> {

    @Query(value = "select case when count(f.id)> 0 then true else false end as
isfollowing from uptake.follower f where f.followed_user_id= :B and
f.following_user_id = :A", nativeQuery = true)
    int isAfollowingB(@Param("A") long A, @Param("B") long B);

    Page<Follower> getAllByFollowedUser(User followingUser, PageRequest pageRequest);

    @Query(value = "delete from uptake.follower where id = :A", nativeQuery = true)
    int unfollowB(@Param("A") long A);
}
```

Follow Service

```
public interface FollowerService {
//    Follower save(User user, User follow);
    Follower addFollow(User followingUser, User followedUser);
    boolean addFollowCheck(User followingUser, User followedUser);
    int isAfollowingB(long A, long B);
    void unfollowB(Long A);
}
```

Follower Service implementation

```
@Service
public class FollowerServiceImpl implements FollowerService {

    private FollowRepository followRepository;

    @Autowired
    public FollowerServiceImpl(FollowRepository followRepository) {
        this.followRepository = followRepository;
    }

    @Override
    public Follower addFollow(User followingUser, User followedUser) {
        return followRepository.save(new Follower(followingUser,followedUser));
    }

    @Override
    public boolean addFollowCheck(User followingUser, User followedUser) {
        System.out.println("AddFollowCheck" + followingUser.getId() + ", " +
followedUser.getId());
        if(followedUser.getId()!=followingUser.getId()){
            if(isAfollowingB(followingUser.getId(),followedUser.getId())==0) {
                followRepository.save(new Follower(followingUser,followedUser));
                return true;
            }
        }
        return false;
    }

    @Override
    public int isAfollowingB(long A, long B) {
        return followRepository.isAfollowingB(A,B);
    }

    @Override
    public void unfollowB(Long A) {
        followRepository.deleteById(A);
    }

//    @Override
//    public Follower save(User user, User follow) {
//        return followRepository.save(new Follower(user, follow));
//    }
}
```

Profile page's part:

```
<div class="col-md-8">
    <div class="relative">
        <strong class="title" th:text="${user.username}"></strong>
        <a th:if="${isMyAccount}" th:text="#{message.profile.editprofile}"
href="/account/edit" class="btn btn-outline-secondary"></a>
        <span th:unless="${isMyAccount}">
            <span th:if="${isFollowing}" class="textFollowing"
th:text="#{message.profile.following}"></span>
            <a th:unless="${isFollowing}" class="follow uptake-btn enabled"
th:href="'/profile/follow/' + ${user.id}" th:data-id="${user.id}" id="anchorFollow"
th:text="#{message.profile.follow}"></a>
            <span  id="textFollowing" class="textFollowing"></span>
        </span>
    </div>
    <div class="attr_wrap mb10 mt20">
        <div class="attr"><span th:text="${user.getPostList().size()}"></span>
posts</div>
        <div class="attr">
            <div th:onclick="'openFollowerModal(\'' + ${user.id} + '\');'">
                <span th:text="${user.getFollowedUsers().size()}"
id="numOfFollower"></span> follower
            </div>
        </div>
        <div class="attr">
            <div th:onclick="'openFollowingModal(\'' + ${user.id} + '\');'">
                <span th:text="${user.getFollowingUser().size()}"></span> following
            </div>
        </div>
    </div>
```

Ajax call from profile

```
$(".follow").click(function (e) {
    e.preventDefault();
    var id = $(this).data("id");
    console.log("Follow called by ajax called" + id);
    $.ajax({
        url: "/profile/followbyajax/" + id,
        success: function (data) {
            var obj = JSON.parse(data);
            if(obj.result){
                $("#numOfFollower").text(obj.n);
                $("#anchorFollow").hide(300);
                $("#textFollowing").text("Following");
            }else{
                console.log("Already following");
            }
        }
    });
});
```

Follow and Unfollow method in FollowController

```
@RequestMapping(value="profile/followbyajax/{userId}", method = RequestMethod.GET)
@ResponseBody
public String ajaxFollow(@PathVariable long userId, Model model) {
    long currentUserId = 1;
    String email = null;
    User loggedUser = null;
    Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (principal instanceof UserDetails){
        email = ((UserDetails) principal).getUsername();
        loggedUser = userService.getUserByEmail(email);
    }

    if(loggedUser != null ){
        currentUserId = loggedUser.getId();
    }
    User followingUser = userService.getUserById(currentUserId);
    User followedUser = userService.getUserById(userId);

    JSONObject response = new JSONObject();
    if(followerService.addFollowCheck(followingUser,followedUser)) {
        response.put("result",true);
        response.put("followingUser",currentUserId);
        response.put("followedUser",userId);
        response.put("n", followedUser.getFollowedUsers().size());

    }else{
        response.put("result",false);
    }
    return response.toString();
}

@RequestMapping(value="profile/unfollowbyajax/{id}", method = RequestMethod.GET)
@ResponseBody
public String ajaxUnFollow(@PathVariable long id, Model model) {
    JSONObject response = new JSONObject();
    Long lId = Long.valueOf(id);
    followerService.unfollowB(lId);
    response.put("result",true);
    response.put("n", lId);
    return response.toString();
}
```

Login success handler

This handler redirect CLIENT to Home page and ADMIN to admin dashboard after sucessful login.

1. Inject AuthenticationSuccessHandler bean in Security config.

```
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

        @Bean
        public AuthenticationSuccessHandler myAuthenticationSuccessHandler(){
            return new MySimpleUrlAuthenticationSuccessHandler();
        }

}
```

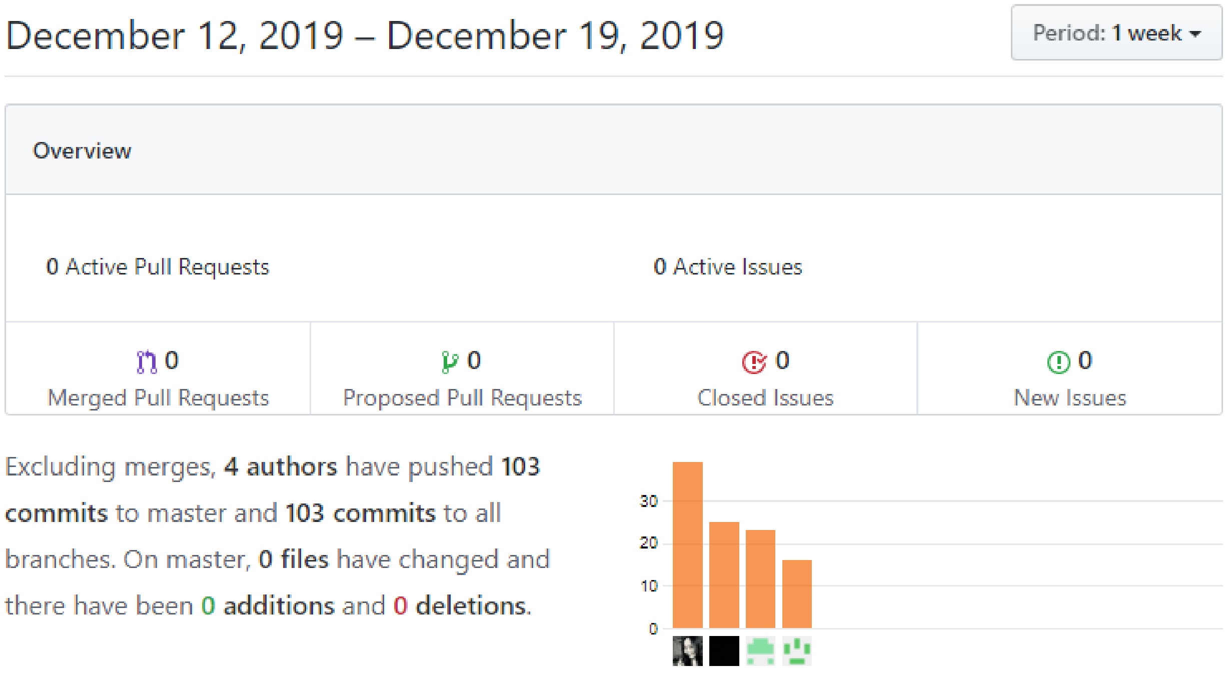
2.Register successHandler in formLogin()

```
@Override
protected void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests()
        .antMatchers("/","/home").authenticated()
        .antMatchers("/postPhoto").authenticated()
        .antMatchers("/postVideo").authenticated()
        .antMatchers("/admin","/admin/**").hasRole("ADMIN")
        .antMatchers("/profile","/profile/**").authenticated()
        .antMatchers("/login").permitAll()
        .antMatchers("/logout").permitAll()
        .and()
        .formLogin()
            .loginPage("/login")
            .successHandler(myAuthenticationSuccessHandler())
            .failureUrl("/login?error=true")
            .usernameParameter("email")
            .passwordParameter("password")
            .and()
        .logout()
        .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
        .clearAuthentication(true)
        .invalidateHttpSession(true)
        .logoutSuccessUrl("/login")
        .and()
        .exceptionHandling().accessDeniedPage("/403Forbidden")
        .and()
        .csrf().disable();

}
```

3. MySimpleUrlAuthenticationSuccessHandler class implementing AuthenticationSuccessHandler

```
public class MySimpleUrlAuthenticationSuccessHandler implements
AuthenticationSuccessHandler {
    protected Log logger = LogFactory.getLog(this.getClass());
    private RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
                                        HttpServletResponse response, Authentication
authentication)
            throws IOException {

        handle(request, response, authentication);
        clearAuthenticationAttributes(request);
    }
    protected void handle(HttpServletRequest request,
                          HttpServletResponse response, Authentication
authentication)
            throws IOException {

        String targetUrl = determineTargetUrl(authentication);

        if (response.isCommitted()) {
            logger.debug(
                    "Response has already been committed. Unable to redirect to "
                            + targetUrl);
            return;
        }

        redirectStrategy.sendRedirect(request, response, targetUrl);
    }
    protected String determineTargetUrl(Authentication authentication) {
        boolean isUser = false;
        boolean isAdmin = false;
        Collection<? extends GrantedAuthority> authorities
                = authentication.getAuthorities();
        for (GrantedAuthority grantedAuthority : authorities) {
            if (grantedAuthority.getAuthority().equals("ROLE_USER")) {
                isUser = true;
                break;
            } else if (grantedAuthority.getAuthority().equals("ROLE_ADMIN")) {
                isAdmin = true;
                break;
            }
        }

        if (isUser) {
            return "/";
        } else if (isAdmin) {
            return "/admin";
        } else {
            throw new IllegalStateException();
        }
    }
    protected void clearAuthenticationAttributes(HttpServletRequest request) {
        HttpSession session = request.getSession(false);
        if (session == null) {
            return;
        }
        session.removeAttribute(WebAttributes.AUTHENTICATION_EXCEPTION);
    }

    public void setRedirectStrategy(RedirectStrategy redirectStrategy) {
        this.redirectStrategy = redirectStrategy;
    }
    protected RedirectStrategy getRedirectStrategy() {
        return redirectStrategy;
    }
}
```

GitHub repository insight



December 12, 2019 – December 19, 2019

Period: **1 week**

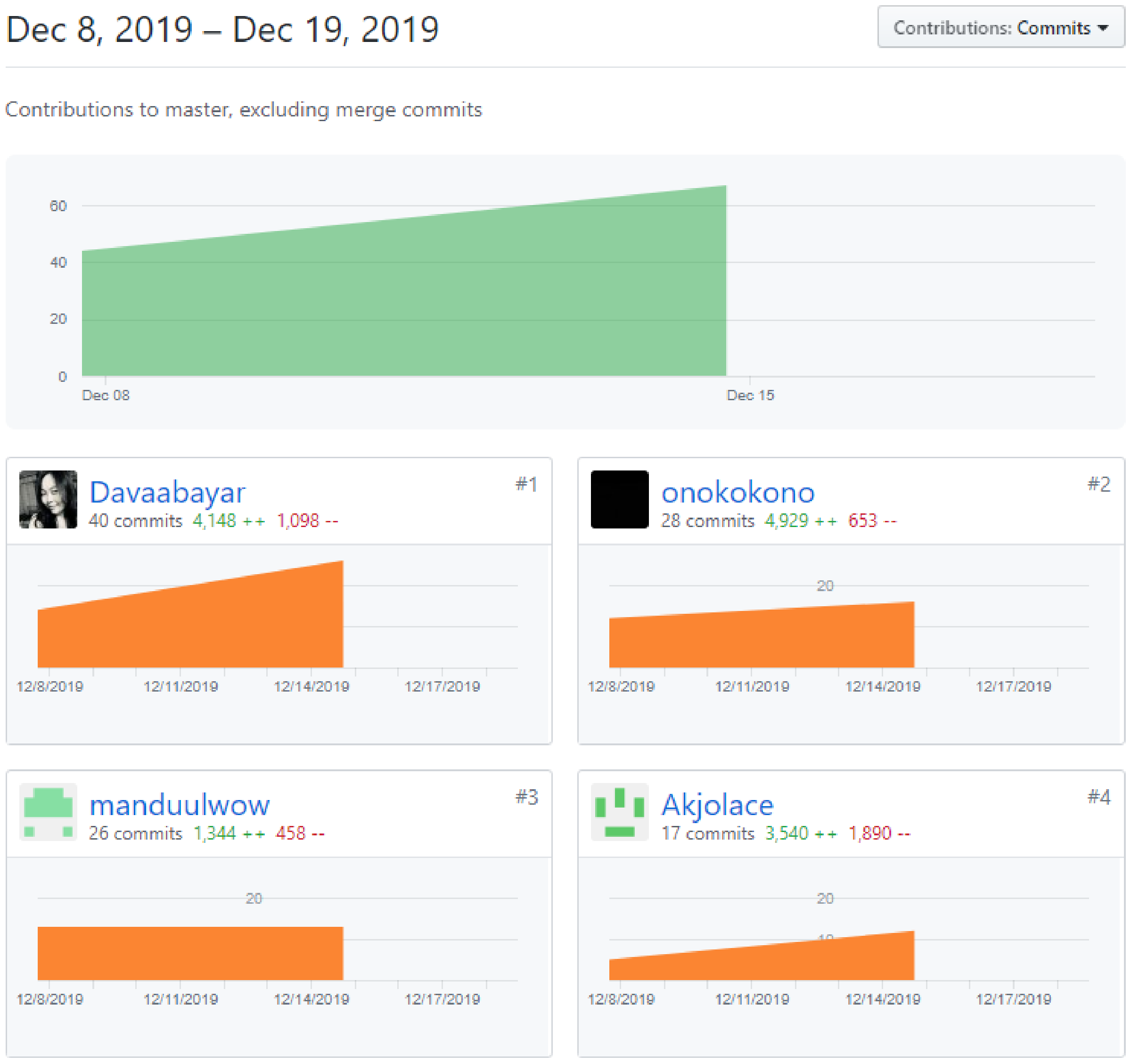
Overview

0 Active Pull Requests

0 Active Issues

0 Merged Pull Requests	**0** Proposed Pull Requests	**0** Closed Issues	**0** New Issues

Excluding merges, **4 authors** have pushed **103 commits** to master and **103 commits** to all branches. On master, **0 files** have changed and there have been 0 **additions** and 0 **deletions**.



Dec 8, 2019 – Dec 19, 2019

Contributions: Commits

Contributions to master, excluding merge commits

Davaabayar #1
40 commits 4,148 ++ 1,098 --

onokokono #2
28 commits 4,929 ++ 653 --

manduulwow #3
26 commits 1,344 ++ 458 --

Akjolace #4
17 commits 3,540 ++ 1,890 --