Ⅰ. Development of flight baggage test service through object detection

1. Purpose

The number of passengers on domestic flights was about 61 million in 2015 to about 90 million in 2019. It has increased about 1.5 times in four years. As the number of people on board increases, the number of people who experience inconvenience in the process of checking their luggage is also increasing. Anyone who has ever used an airplane must have had the experience of carrying things on board and throwing them away just before boarding the plane. In order to avoid this experience, existing airplane passengers wanted to determine whether certain items should be allowed into the aircraft during carrier packing process. To do so, they had to first come up with keywords for the item, then go to a website that offered related services and search for permission. There was inconvenience in going through these two processes, and there were also many items that did not appear in the keyword search. To eliminate this inconvenience, we created an application that tells us whether the item can be brought into the plane through object recognition technology.

2. Period

April 10, 2020 to June 24, 2020

3. Position

This project team consisted of three developers and one mentor. As a project leader, I have contributed to all phases of the project. The project consisted of a total of five stages: Planning and Analysis, Database, Algorithm, UI, and Test. I was the founder of the project and participated in project planning, requirements collection and analysis, and function elicitation during the Planning and Analysis phase. At the Database stage, I was in charge of designing databases and establishing DBs about carry-on restriction by country. Algorithm design and text extraction functions were implemented in the algorithm stage as well, and app UI design and composition were taken charge in the UI stage. In the test phase, I was in charge of testing about the lowest SDK version (24 SDK) set in the project and conducted the evaluation and modification of the test with our team mentor.

4. Skills

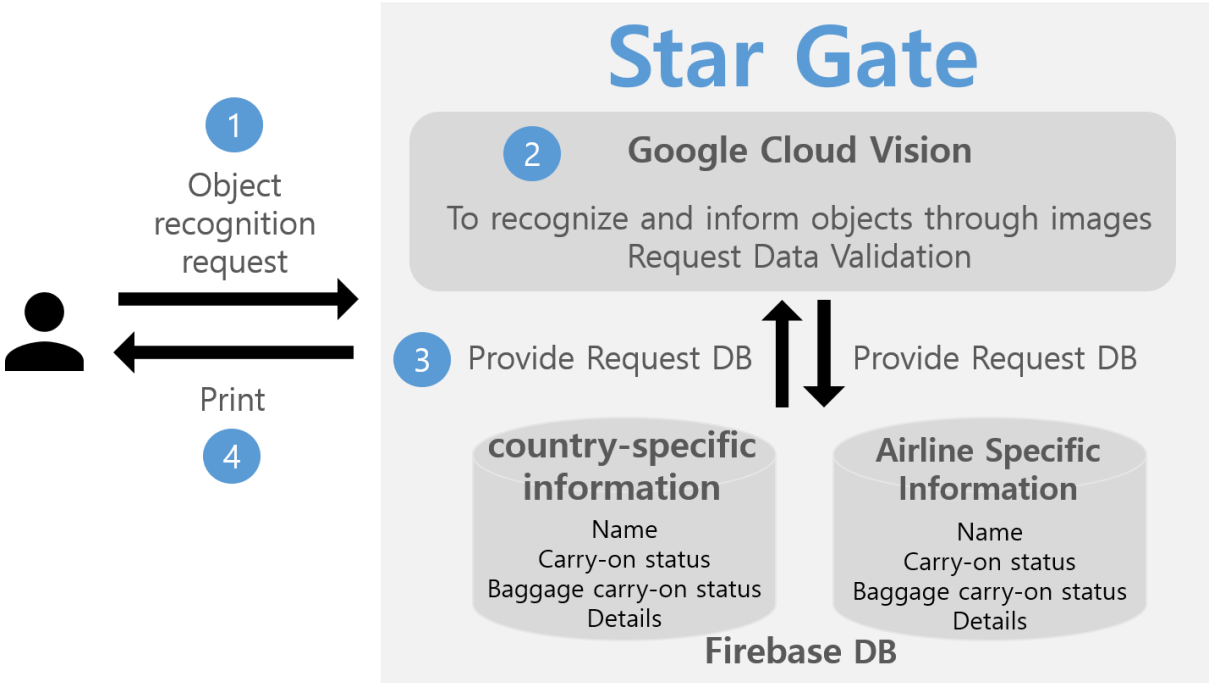
Before entering the project design phase, we defined requirements, UseCase diagrams and system schemes to further specify the project. This was organized on Wiki to enable storage and sharing. In addition, the WBS structure chart, Gantt chart, and expected UI were created to efficiently carry out the project.

First, at the project design stage, we compared the recognition rate and reliability of the Tensor Flow and Cloud Vision API to provide high-quality services to users, and as a result, we implemented the object recognition function using the Cloud Vision API. Next, for real-time processing, we used Firebase to create a database for each country and airline for carry-on. The UI has configured the top bar for user feedback, country and airline settings, and the bottom bar for how to search for cameras, galleries, and keyboards.

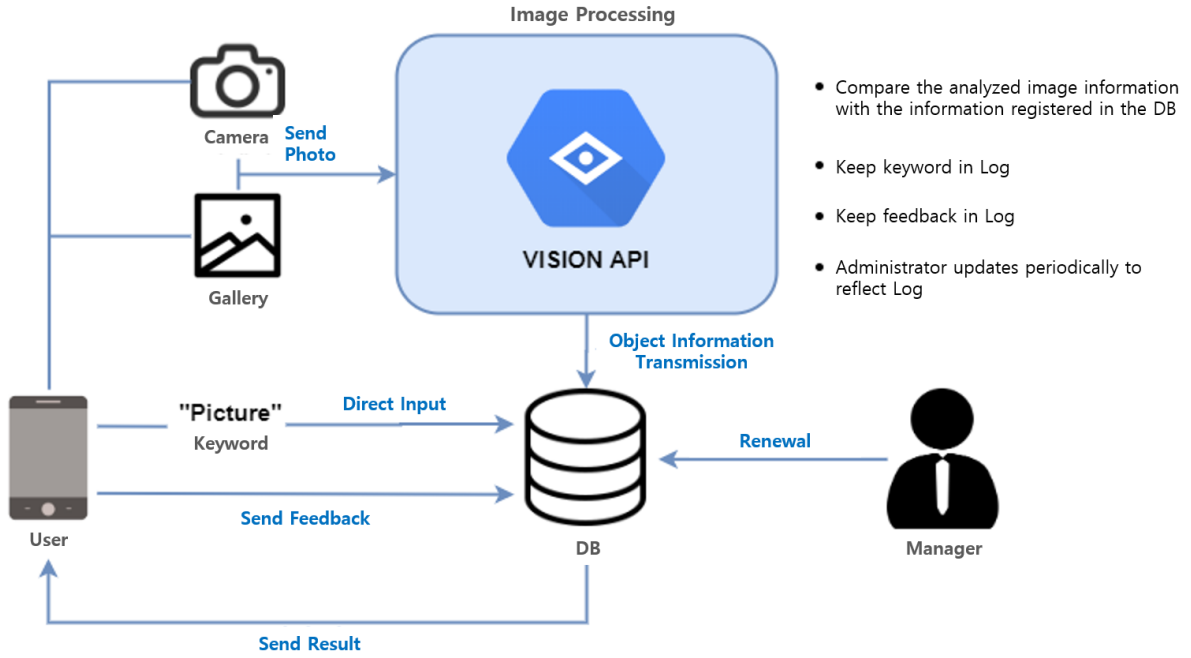
We developed the app by Java through the Android studio. In the actual development stage, Github, Slack, and wiki were used to work together with team members. Development meeting was conducted using Slack, and developers developed together using Github. In particular, it was used as a venue for sharing works such as pushing Github and sharing documents through Slacks.

5. Implement method and Algorithm

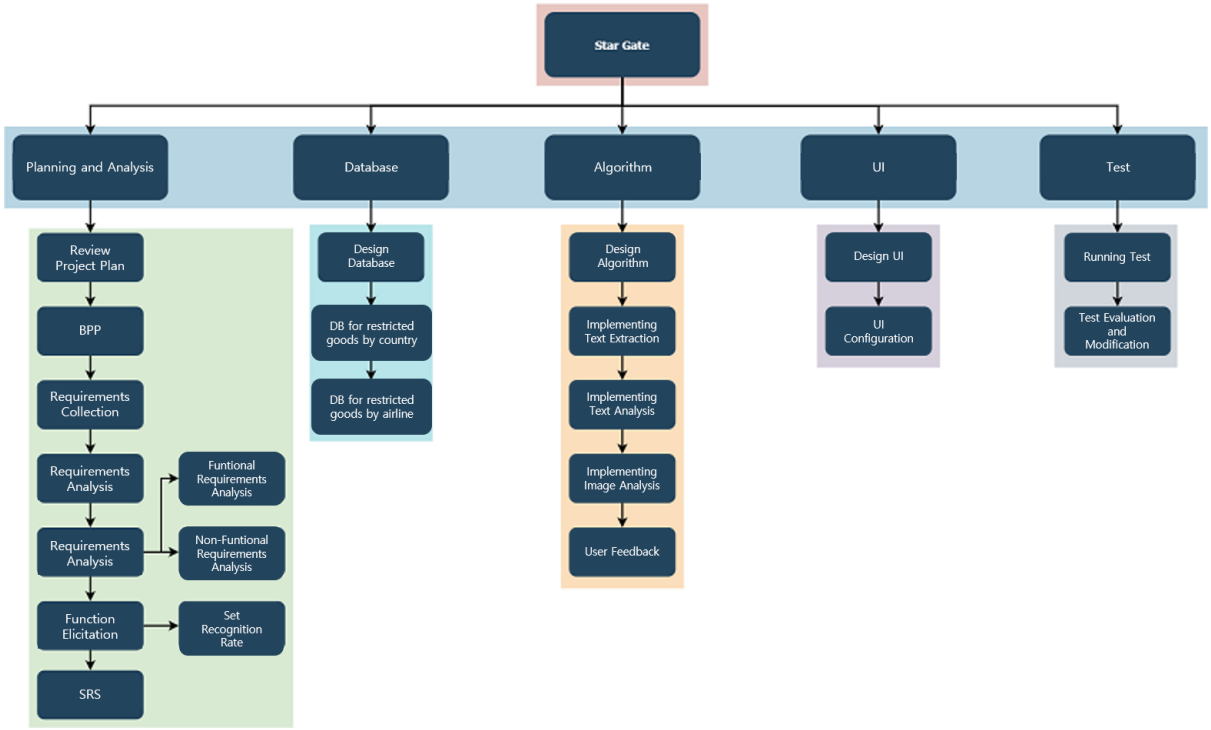
1) System Scenario



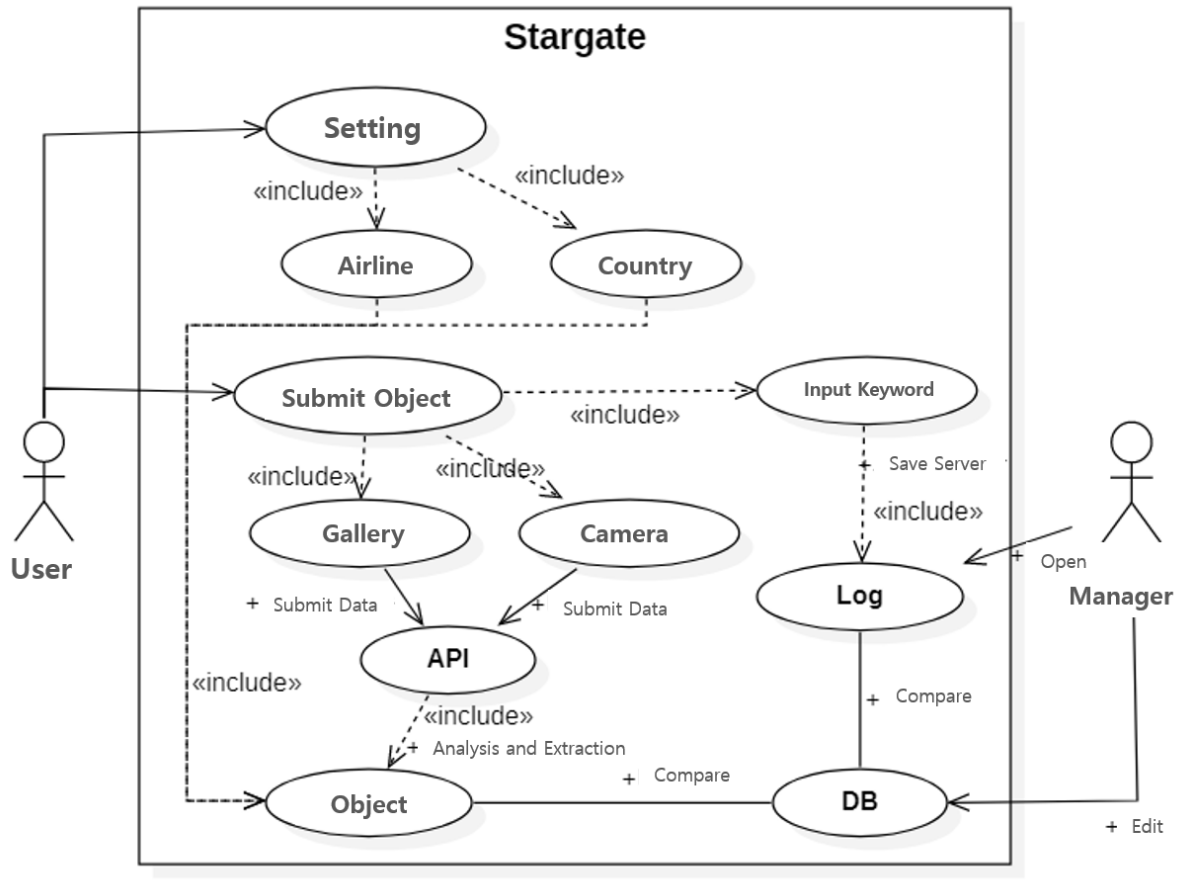
2) System Structure



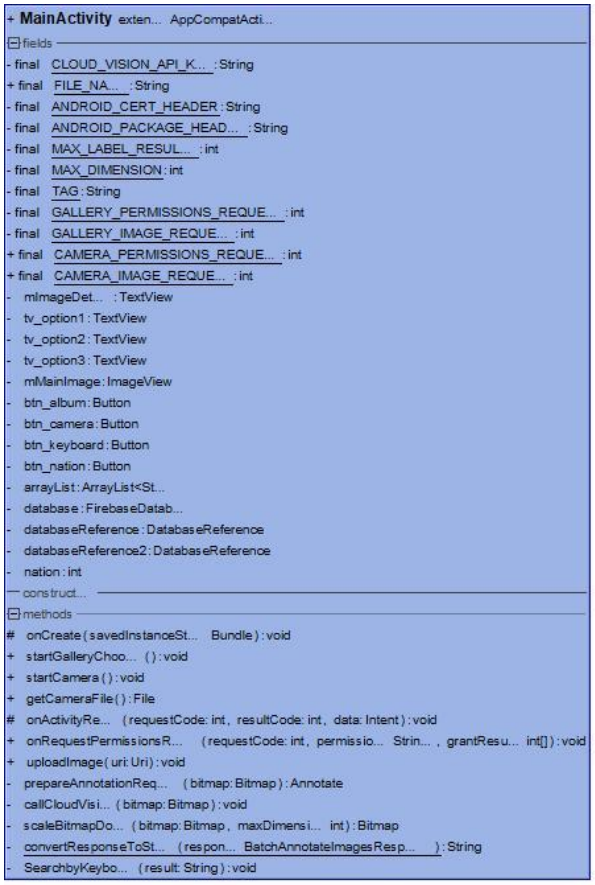
3) WBS

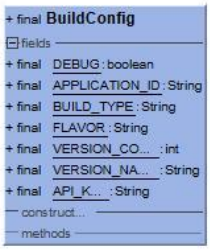


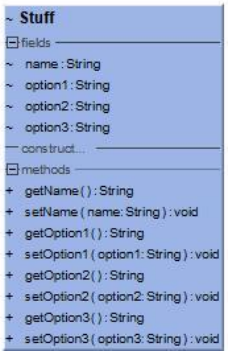
4) UseCase Diagram



5) UML Diagram







(1) Class Specification

① Class MainActivity

Class Name	MainActivity
Class Type	Class
Class Overview	Manage the overall operation of the application.
Parent Class	
Attributes	
- CLOUD_VISION_API_KEY: String + FILE_NAME: String - ANDROID_CERT_HEADER: String	

- ANDROID_PACKAGE_HEADER: String

- MAX_LABEL_RESULTS: int

- MAX_DIMENSION: int

- TAG: String

- GALLERY_PERMISSIONS_REQUEST: int

- GALLERY_IMAGE_REQUEST: int

+ CAMERA_PERMISSIONS_REQUEST: int

+ CAMERA_IMAGE_REQUEST: int

- mImageDetails, tv_option1, tv_option2, tv_option3: TextView

- mMainImage: ImageView

- btn_album, btn_camera, btn_keyboard, btn_nation: Button

- arrayList: ArrayList<Stuff>

- database: FirebaseDatabase

- databaseReference, databaseReference2: DatabaseReference

- nation: int

Operations

onCreate(Bundle savedInstanceState): void
// Create UI and associate DB with users.

+ startGalleryChooser(): void
// Obtain permission to retrieve the gallery's photo information and associate it with the user's gallery.

+ startCamera(): void
// Obtain permission to shoot the camera and connect the camera with the user.

+ getCameraFile(): File
// Bring the pictures taken with the camera.

onActivityResult(int requestCode, int resultCode, Intent data): void
// Uploads object information from the user-supplied image forms.

+onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults): void
// Receive access to the user's chosen method of object information delivery.
 	
+ uploadImage(Uri uri): void
// An image processed to suit the form is provided to the analysis algorithm.
 	
- prepareAnnotationRequest(Bitmap bitmap): Annotate
// Connect the image analysis algorithm.
 	
- callCloudVision(Bitmap bitmap): void
 	
// Analyze the entered image.
 	
- scaleBitmapDown(Bitmap bitmap, int maxDimension): Bitmap
 	
// Resize the provided image to fit the form.
 	
- convertResponseToString(BatchAnnotateImagesResponse response): String
// Transmits the response to the image into a String format.
 	
- SearchbyKeyboard(String result): void
 	
// Object information corresponding to the keyword entered is retrieved from the DB.

② Class PackageManagerUtils

Class Name	PackageManagerUtils
Class Type	Class
Class Overview	Perform a utility to obtain SHA1 signatures for the application.
Parent Class	
Attributes	
Operations	
+ getSignature(@NonNull PackageManager pm, @NonNull String packageName): String // Obtain encryption signatures to work with Google CloudVision APIs. - signatureDigest(Signature sig) : String	

// To process a signature into a fixed form.

③ Class PermissionUtils

Class Name	PermissionUtils
Class Type	Class
Class Overview	Perform a utility to obtain permission other than the app.
Parent Class	
Attributes	
Operations	

+ requestPermission(Activity activity, int requestCode, String... permissions): boolean
// Request non-app permissions and return results.
+ permissionGranted(int requestCode, int permissionCode, int[] grantResults): boolean
// Return information to request non-app permissions.	

④ Class Stuff

Class Name	Stuff
Class Type	Class
Class Overview	Form of information stored in DB
Parent Class	
Attributes	
- name: String - option1: String - option2: String - option3: String	

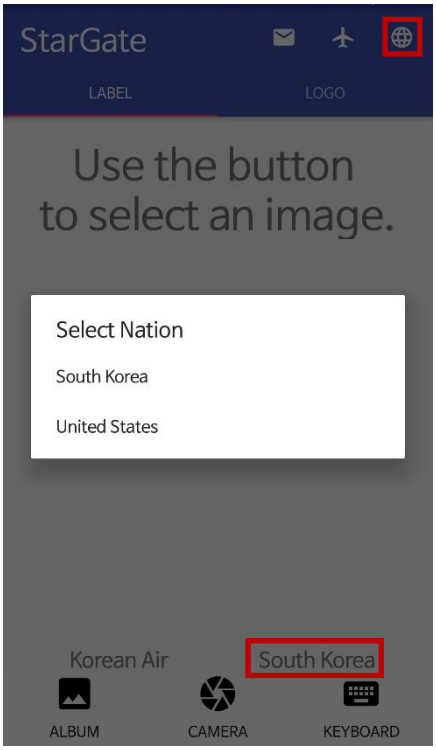
Operations

+ getName(): String
 	
// Get name.
 	
+ setName(String name): void
// Register the name received.
 	
+ getOption1(): String
 	
// Get information about carry-on.
 	
+ getOption2(): String
 	
// Get information about checked baggage.
 	
+ getOption3(): String
 	
// Get other information.

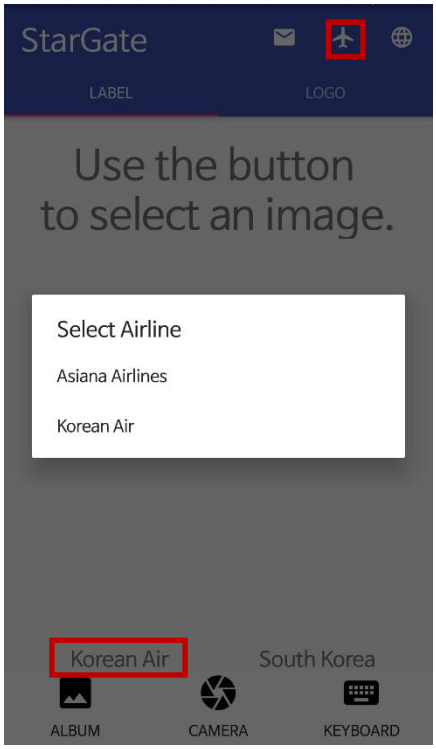
6) UI

Main

Select Nation

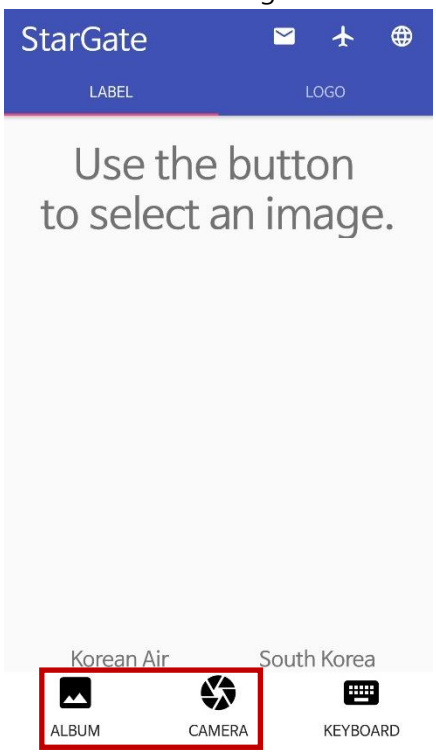


Select Airline



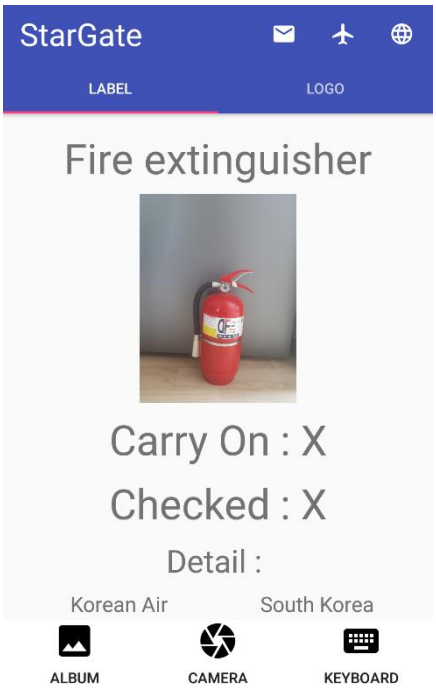
Select LABEL/LOGO

Send Image

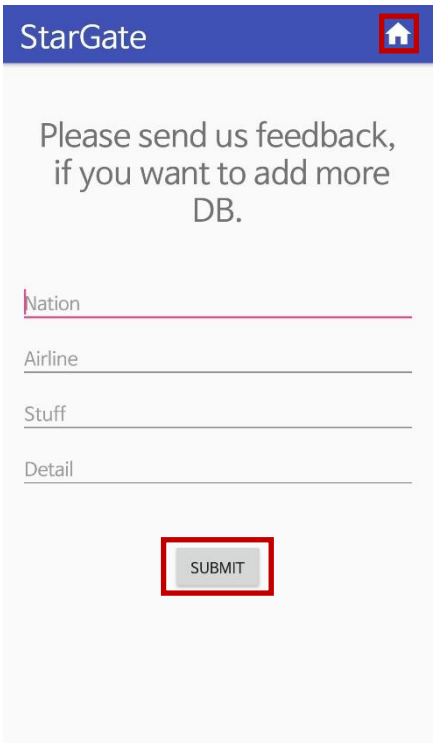


Send Keyword

Check Result



User Feedback



Ⅱ. Bundes Report

1. Purpose

Today, we continue to have special communication that shares our daily lives, empathizes with each other, and creates new things that we need. Our communication connects each other, creates new values, and supports you who love Germany and Korea. So that the excitement of starting a new life on the other side of the globe and the joy of free communication continue every single day,

"Bundes Report is in your daily life today."

2. Period

November 10, 2020 to June December 02, 2020

3. Position

We started our project during K-Move(Germany) IT School. This project team consisted of three developers. I was the only Front-End developer on this team. I had full responsibility for the view and layout. In addition, to increase user revisit rates and to help learning German, I created game corners.

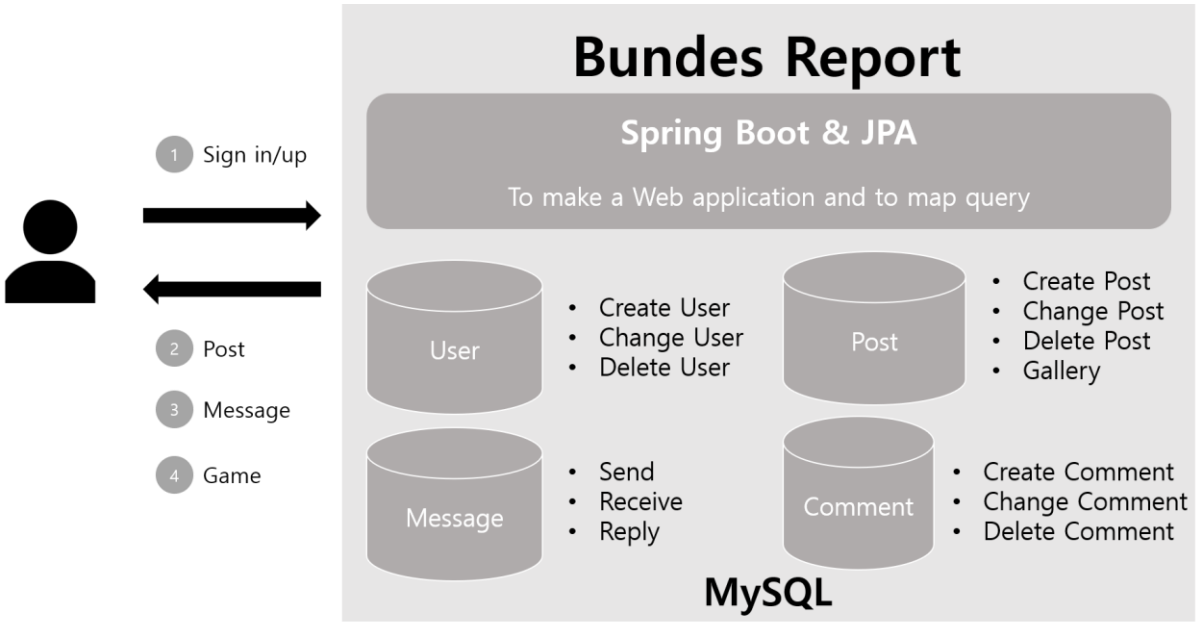
4. Skills

Our web project started from the Java project and this project based on Java Spring Boot and JPA. JPA was main data processing technology of our team. For Game Function, I used also JPA to map our query. Thanks to JPA, we didn't need to make a SQL query.

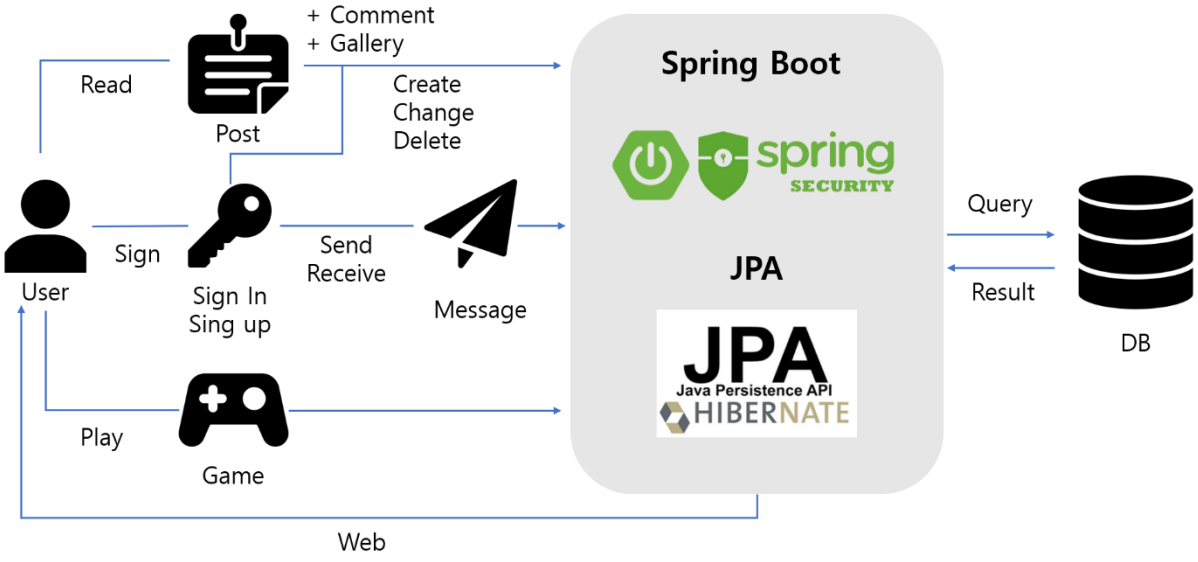
My main job was view and layout. We used Thymeleaf as a template engine. On the client side, code was written in HTML format and was used to dynamically draw DOMs. It was responsible for the process of receiving data and dynamically drawing that on DOM objects.

5. Implement method and Algorithm

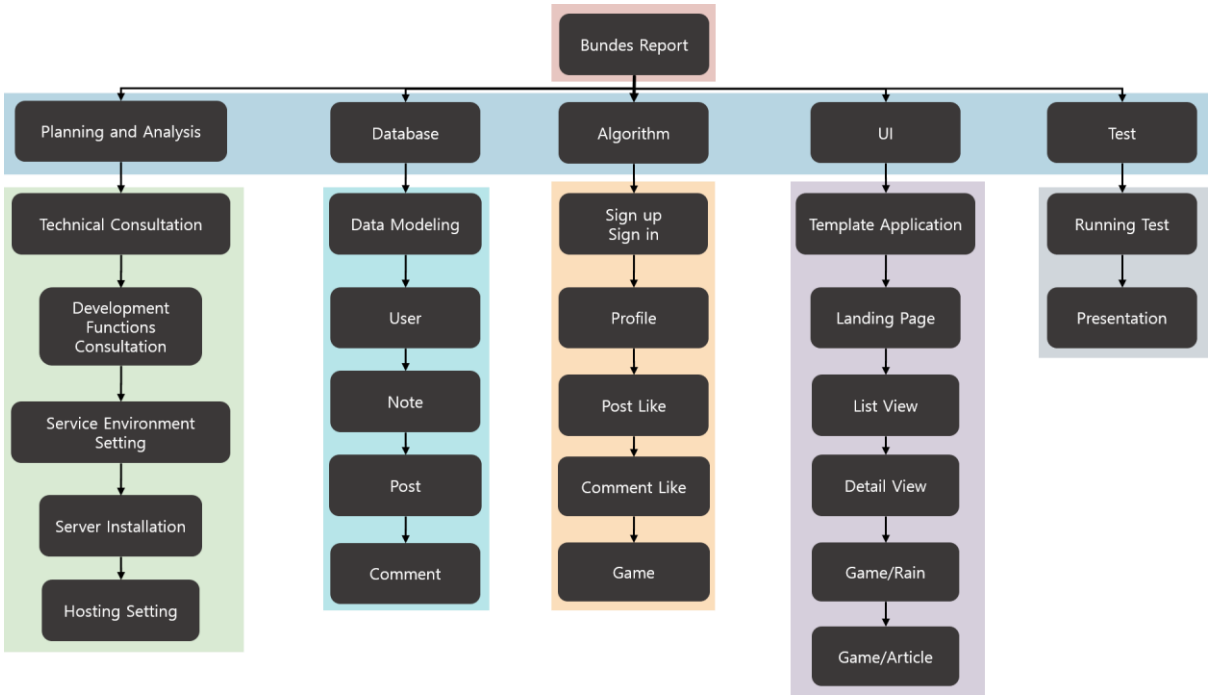
1) System Scenario



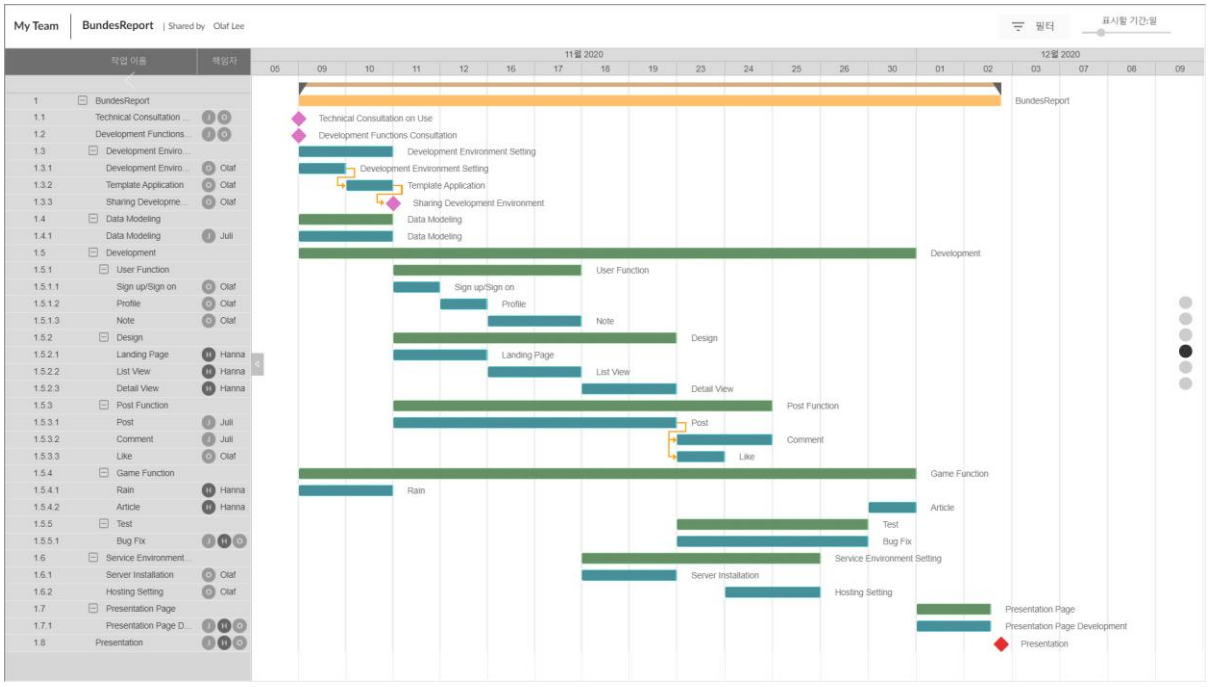
2) System Structure



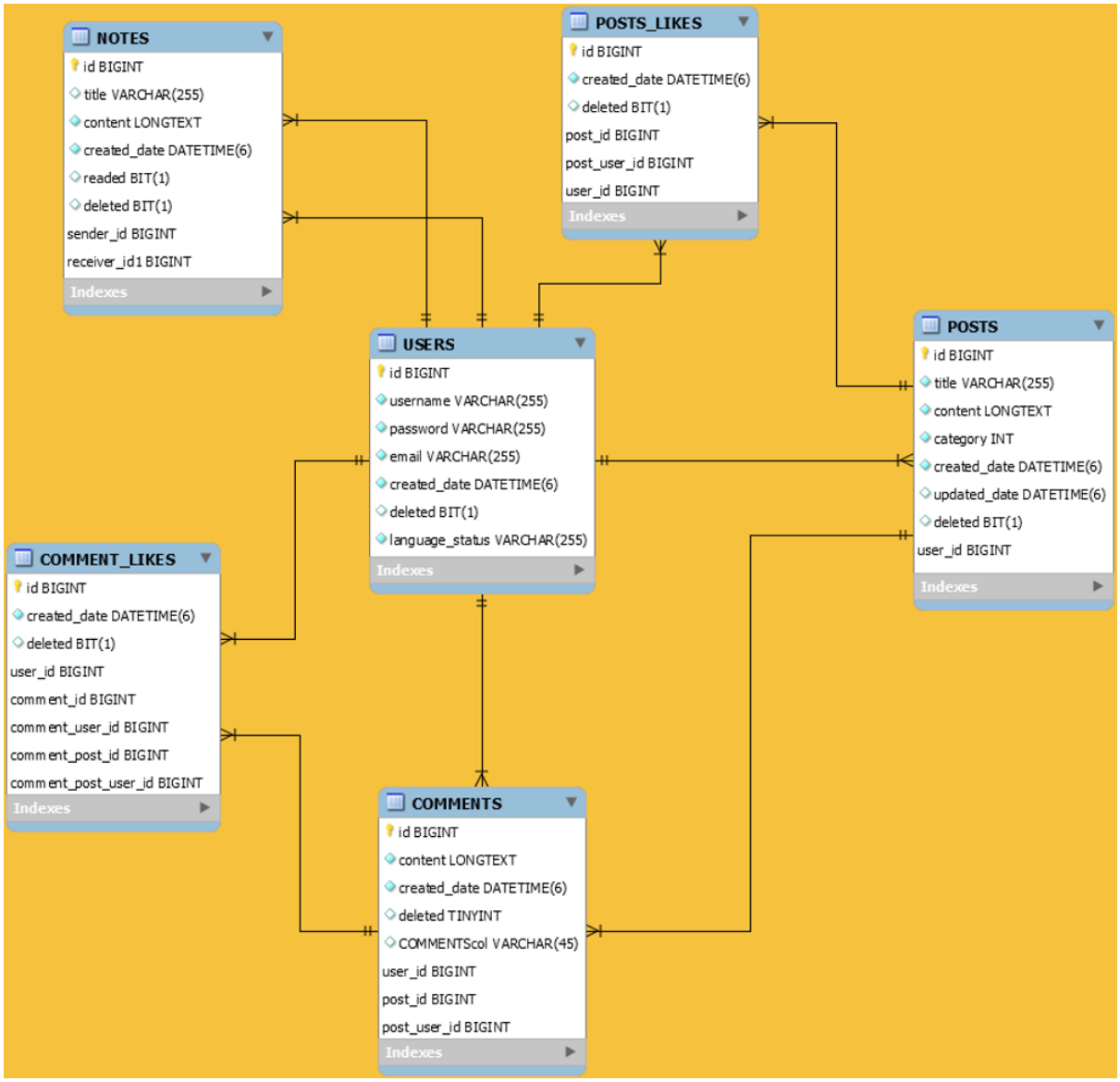
3) WBS



4) Gant Chart



5) UML Diagram



(1) Class Specification

① Class User

Class Name	User
Class Type	Class
Class Overview	Create User Model
Parent Class	
Attributes	

- serialVersionUID: long

// Serialization

- id: Long

// Primary Key

- deleted: String

- username: String

- password: String

- email: String

- createdDate: LocalDateTime

// Save time automatedly when insert

- languageStatus: LanguageStatus

// Enum [KO, DE]

- posts: List<Post>

// Mapped by User

- postLikes: List<PostLike>

// Mapped by User

- comments: List<Comment>

// Mapped by User

- commentLikes: List<CommentLike>

// Mapped by User

- sendNotes: List<Note>

// Mapped by User

- receiveNotes: List<Note>

Operations

+ User(Long id, boolean deleted, String username, String password, String email, LocalDateTime createdDate, LanguageStatus languageStatus)
 	
// Create Entity Object
 	
+ toUserForm(): UserForm
 	
// Create User insert form
 	
+ getAuthorities(): Collection<? extends GrantedAuthority>
 	
// Spring Security Authority
 	
+ getUsername(): String
 	
+ isAccountNonExpired(): boolean
 	
+ isCredentialsNonExpired(): boolean
 	
+ isEnabled(): boolean

② Class Note

Class Name	Note
Class Type	Class
Class Overview	Create Note Model
Parent Class	
Attributes	

- id: Long
 	
// Primary Key
 	
- deleted: boolean
 	
- title: String
 	
- content: String
 	
- createdDate: LocalDateTime
 	
// Save time automatedly when insert	

- readed: Boolean
- sender: User
// Join with sender_id
- receiver: User
// Join with receiver_id

Operations
+ Note(Long id, boolean deleted, String title, String content, LocalDateTime createdDate, boolean readed, User sender, User receiver) // Create Entity Object

③ Class Post

Class Name	Post
Class Type	Class
Class Overview	Create Post Model
Parent Class	
Attributes	

- id: Long
// Primary Key
- deleted: boolean
- title: String
- content: String
- category: CategoryType
- createdDate: LocalDateTime
// Save time automatedly when insert
- user: User	

// Join with user_id
- comments: List<Comment>
// Mapped by post
- likes: List<PostLike>
// Mapped by post
- viewCount: int

Operations

+ Post(Long id, boolean deleted, String title, String content, CategoryType category, LocalDateTime createdDate, User user, int viewCount)
// Create Entity Object
+ toPostForm(): PostForm
// Create Post insert form
+ getUpdateModel(PostForm form): Post
// Get Title and Content from form
+ getLikeCount(): int
+ getCommentCount(): int
+ getFirstImage(): String
// show no-image when there is no image in gallery

④ Class PostLike

Class Name	PostLike
Class Type	Class
Class Overview	Count Like in Post
Parent Class	
Attributes	

- id: Long
// Primary Key
- deleted: boolean
- user: User
// Join with user_id
- post: Post
// Join with post_id
- createdDate: LocalDateTime
// Save time automatedly when insert

Operations
+ PostLike(Long id, boolean deleted, User user, Post post, LocalDateTime createdDate) // Create Entity Object

⑤ Class Comment

Class Name	Comment
Class Type	Class
Class Overview	Create Comment model
Parent Class	
Attributes	

- id: Long
// Primary Key
- deleted: boolean
- content: String
- createdDate: LocalDateTime
// Save time automatedly when insert	

- user: User
// Join with user_id
- post: Post
// Join with post_id
- likes: List<CommentLike>
// Mapped by comment

Operations

+ Comment (Long id, boolean deleted, String content, LocalDateTime createdDate, User user, Post post)
// Create Entity Object
+ toCommentForm(): CommentForm
// Create Comment insert form
+ getUpdateModel(CommentForm form): Comment
// Get Content from form
+ hasMineLike(User user): boolean
// Check pressed Like
+ getLikeCount(): int

⑥ Class CommentLike

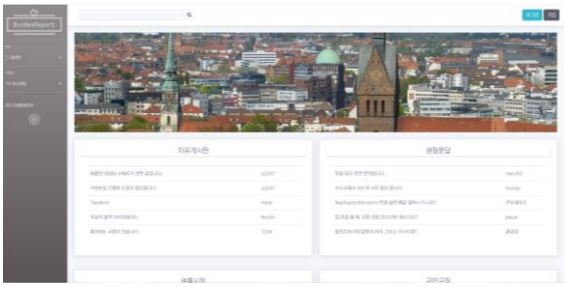
Class Name	CommentLike
Class Type	Class
Class Overview	Count Like in Comment
Parent Class	
Attributes	
- id: Long	

// Primary Key
 	
- deleted: Boolean
 	
- user: User
 	
// Join with user_id
 	
- comment: Comment
 	
// Join with comment_id
 	
- createdDate: LocalDateTime
 	
// Save time automatedly when insert

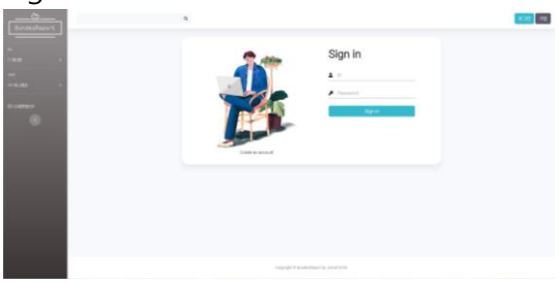
Operations
+ CommentLike (Long id, boolean deleted, User user, String content, LocalDateTime createdDate) // Create Entity Object

6) UI

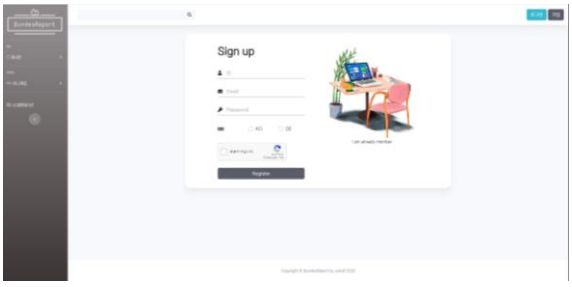
Main Page



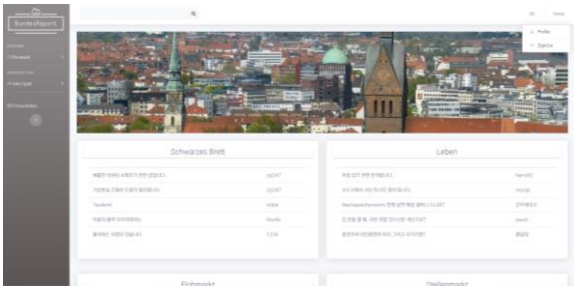
Sign in



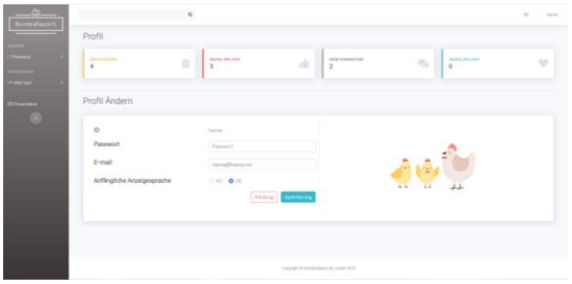
Sign up



Profile Menu Expand

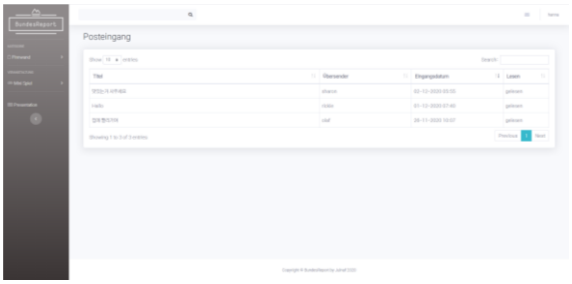


Profile

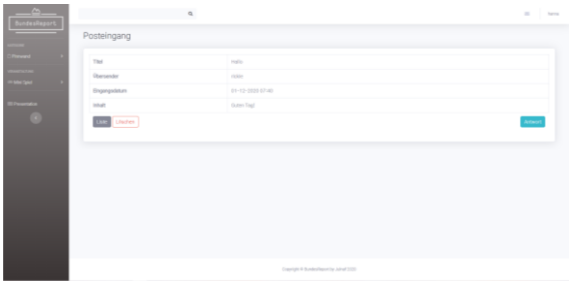


Message Center

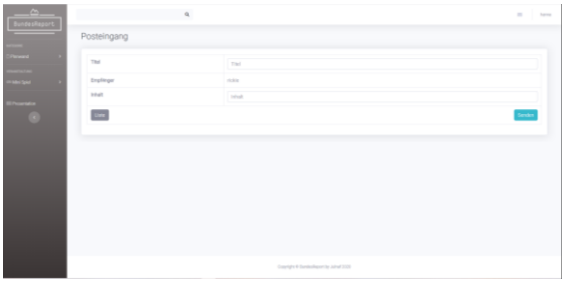
Message List



Message View

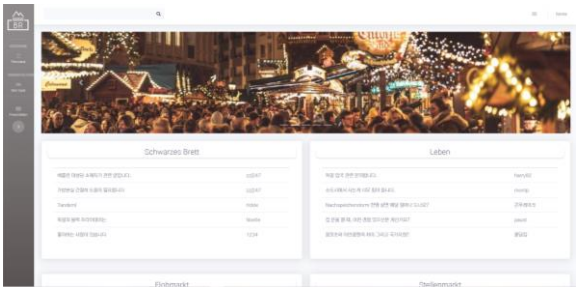


Message Send Form

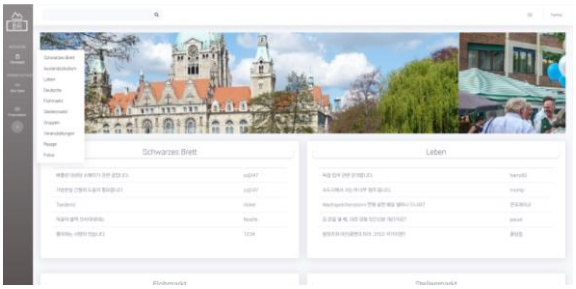


Menu Expand

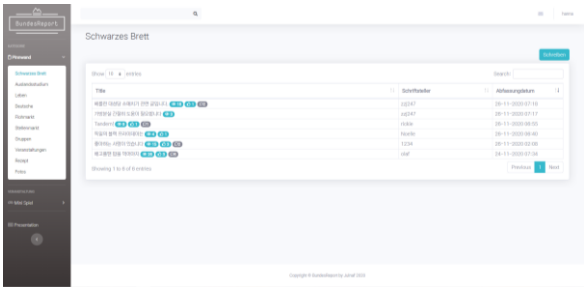
Small Menu



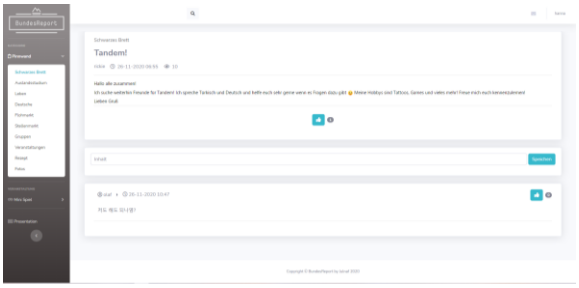
Small Menu Expand



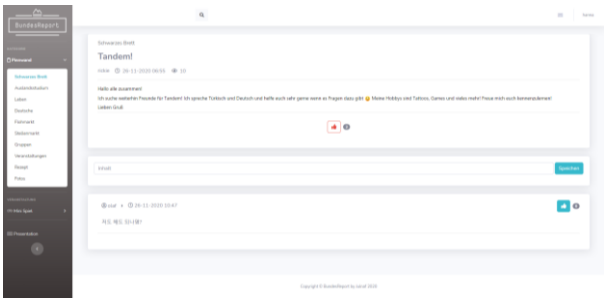
Post List



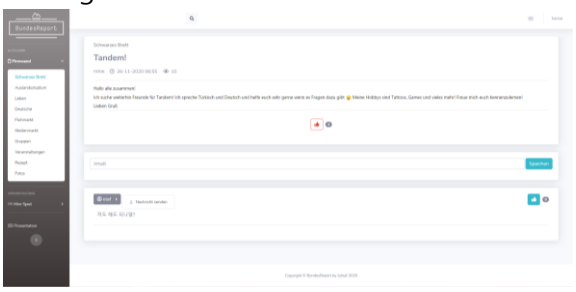
Post View



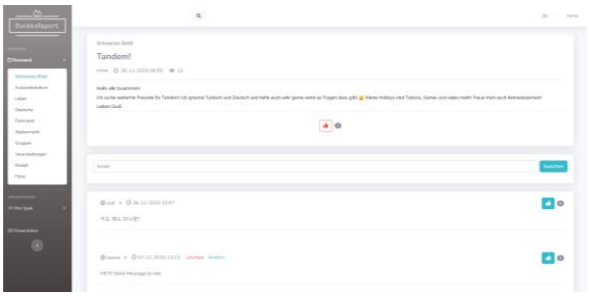
Pressed Like



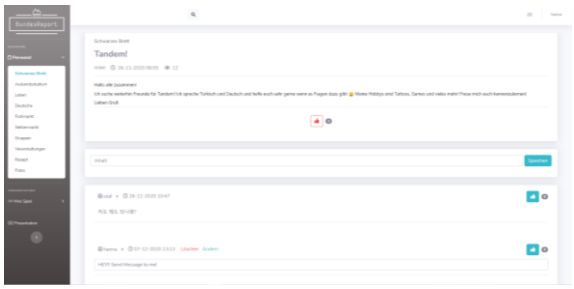
Message Send Menu



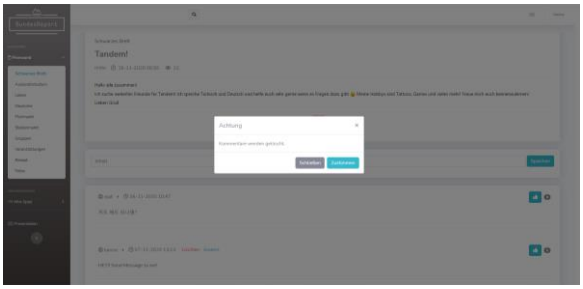
Writted Comment



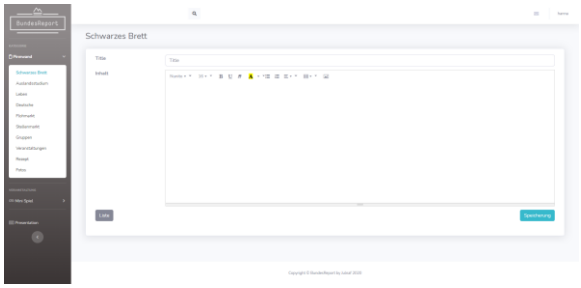
Change Comment



Delete Comment Modal

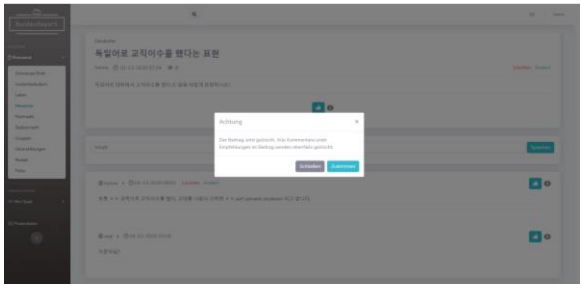


Write Post Form

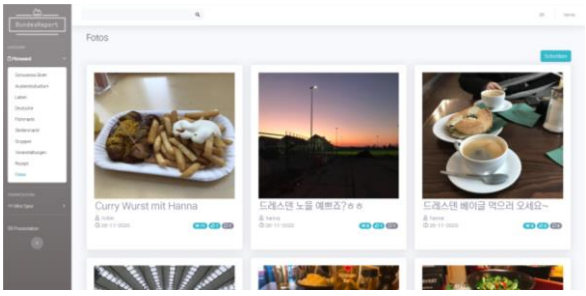


Writted Post

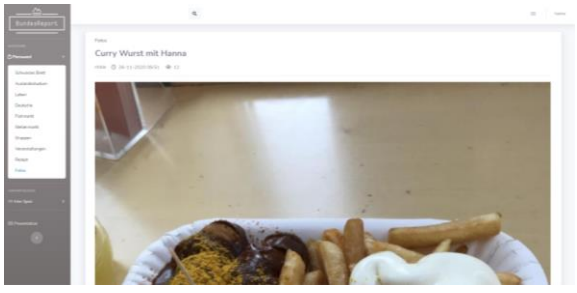
Delete Post Form



Gallery List



Gallery View



Rain Game

Article Game