# CHAPTER 3: PROJECT ORGANIZATION

The success of any software development project is deeply rooted in its organization and structure. A well-structured project ensures seamless interaction between all components, promoting efficiency and reducing the likelihood of errors. It also facilitates easier maintenance and scalability, allowing for the addition of new features or modifications with minimal disruption to existing functionalities.

In this chapter, we delve into the organization and structure of the CommuniView system. The organization and structure of the CommuniView system are influenced by several factors, including the project requirements, the chosen architectural pattern, the design considerations, and the technology stack. Each of these factors plays a crucial role in shaping the system's structure and determining how different components of the system interact with each other. By the end of this chapter, you will have a comprehensive understanding of the CommuniView system's organization and structure, and how its various components work together to provide a platform for users to find and support local businesses.

## 3.1 Project Requirements

The CommuniView web application is designed to cater to three main actors: Users, Business owners, and Admins. Each actor interacts with the system in unique ways, which we will detail in the following use cases. Each use case provides a detailed description of the flow of events, participating actors, entry and exit conditions, and quality requirements.

### 3.1.1 Use Case 1: Register as a New User

This use case is initiated by a user who wants to create a new account on the CommuniView platform. The user accesses the CommuniView web app and selects the option to register as a new user. The system presents a registration form for the user to enter their details. Once the user submits the form, the system processes the registration and creates a new user account. The entry condition for this use case is that the user is on the CommuniView web app, and the exit conditions are either that the user has successfully registered and logged in to their account or that the user decides not to register and navigates away from the registration form. The quality requirements for this use case are that the registration process should be secure and user-friendly, and the registration form should validate the entered information and provide clear error messages.

### 3.1.2 Use Case 2: Search for a Local Business

This use case is also initiated by a user who wants to find a local business on the CommuniView platform. The user enters a search query (category or location) in the search bar, and the system processes the query and displays a list of relevant local businesses. The user can then click on a business listing to view more details. The entry condition for this use case is that the user is on the CommuniView web app, and the exit conditions are either that the user has found the desired local business and viewed its details or that the user decides to perform a new search or leaves the platform. The quality requirements for this use case are that the search results should be accurate and relevant to the user's query, the search results should be displayed within a few seconds, and the search functionality should include filters and sorting options to help users refine their search.

### 3.1.3 Use Case 3: Explore Businesses by Trending Tags

This use case begins when a user notices a section displaying trending tags or categories based on popular searches or recent activity. The user clicks on a trending tag to view businesses associated with that category, and the system displays a list of local businesses related to the selected tag. The entry condition for this use case is that the user is on the CommuniView web app, and the exit conditions are either that the user has explored businesses under the selected trending tag or that the user decides to explore other tags or leaves the platform. The quality requirements for this use case are that trending tags should be updated dynamically based on user activity, the trending tags section should be visually clear and engaging, and users should have the option to easily navigate back to the homepage or switch to other tags.

### 3.1.4 Use Case 4: Discover Special Offers

This use case is initiated by a user who wants to explore discounts in the "Special Offers" section. The user navigates to this section, either on the homepage or through the navigation menu. The system then displays a curated list of local businesses offering special promotions, discounts, or exclusive deals. The user can click on a business offering a special deal to view more details, and the system provides information about the special offer, including terms and conditions. The user can then decide to take advantage of the special offer either by visiting the business in person or using a provided online code. If applicable, the user can leave a review and rating based on their experience with the special offer. The entry condition for this use case is that the user is on the CommuniView web app, and the exit conditions are either that the user has explored special offers and possibly redeemed one or that the user decides to explore other features or leaves the platform. The quality requirements for this use case are that the Special Offers section should be regularly updated with accurate and current promotions, businesses should have an easy process to update their special offers on the platform, users should find the special offers visually appealing and easily accessible, and users should have the option to easily navigate back to the homepage or switch to other tags.

### 3.1.5 Use Case 5: Leave a Review and Rating

This use case begins when a user accesses the CommuniView web app and searches for a local business. The user clicks on a business listing to view its details and selects the option to leave a review and rating for the business. The system presents a form for the user to enter their review and select a rating. The user submits the review and rating, and the system processes the submission and updates the business listing with the new review and rating. The entry condition for this use case is that the user is viewing a local business's details on the CommuniView web app, and the exit conditions are either that the user has successfully submitted a review and rating for the local business or that the user decides not to submit a review and rating and navigates away from the form. The quality requirements for this use case are that the review submission process should be user-friendly and intuitive, the submitted review and rating should be displayed on the business listing within a reasonable time, and the review form should include options for users to upload photos of their experience at the business.

### 3.1.6 Use Case 6: Bookmark a Local Business

This use case is initiated by a user who wants to bookmark a local business for future reference. The user accesses the CommuniView web app and searches for a local business. The user clicks on a business listing to view its details and selects the option to bookmark the business. The system then adds the business to the user's list of bookmarked businesses. The entry condition for this use case is that the user is viewing a local business's details on the CommuniView web app, and the exit conditions are either that the user has successfully bookmarked the local business or that the user decides not to bookmark the business and navigates away from the business details page. The quality requirements for this use case are that the bookmarking process should be user-friendly and intuitive, and the bookmarked businesses should be easily accessible for the user.

### 3.1.7 Use Case 7: Claim a Business Listing

This use case is initiated by a business owner who wants to claim their business listing. The business owner accesses the CommuniView web app and searches for their business. The business owner clicks on their business listing to view its details and selects the option to claim the business listing. The system presents a form for the business owner to provide verification information. The business owner submits the verification information, and the system processes the submission and, upon successful verification, grants the business owner control over the business listing. The entry condition for this use case is that the business owner is viewing their business's details on the CommuniView web app, and the exit conditions are either that the business owner has successfully claimed their business listing or that the business owner decides not to claim the business listing and navigates away from the form. The quality requirements for this use case are that the business claiming process should be secure and reliable, and the verification process should be completed within a reasonable time.

### 3.1.8 Use Case 8: Update Business Information

This use case is initiated by a business owner who wants to update business information on their business listing. The business owner accesses the CommuniView web app and logs in to their account. The business owner navigates to their claimed business listing and selects the option to update the business information. The system presents a form for the business owner to edit the business information. The business owner makes the necessary changes and submits the updated information. The system processes the submission and updates the business listing with the new information. The entry condition for this use case is that the business owner is logged into their account on the CommuniView web app and has claimed their business listing, and the exit conditions are either that the business owner has successfully updated their business information or that the business owner decides not to update the business information and navigates away from the form. The quality requirements for this use case are that the business information update process should be user-friendly and intuitive, the updated information should be displayed on the business listing within a reasonable time, and the business owner should be able to add or update photos, business hours, and special offers.

### 3.1.9 Use Case 9: Monitor and Manage the Platform

This use case is initiated by an admin who wants to monitor and manage the CommuniView platform. The admin accesses the CommuniView web app and logs in to their account. The admin navigates to the platform management dashboard and takes necessary actions, such as removing inappropriate content or addressing user concerns. The entry condition for this use case is that the admin is logged into their account on the CommuniView web app, and the exit conditions are either that the admin has completed their monitoring and management tasks or that the admin logs out of their account or navigates away from the platform management dashboard. The quality requirements for this use case are that the platform management dashboard should provide comprehensive and relevant information for the admin, and the admin should be able to efficiently address issues and manage the platform. The use case diagram in Figure 3.1 provides a visual representation of these use cases and their relationships with the actors. In the next sections, we will delve into the architecture, design, and technology stack of the CommuniView system.

**A close-up of a chat box

Description automatically generated**

Figure 3.1: UML use case diagram for CommuniView.

## 3.2 Architecture

The architecture of a software system plays a crucial role in determining its success. A well-defined architecture provides a blueprint for the system, outlining its components and the relationships between them. It also sets the foundation for the system's design, implementation, and maintenance.

In the case of the CommuniView system, the architecture is defined by the entity, boundary, and control objects. These objects form the building blocks of the system, each with a distinct role and set of attributes. They also interact with each other in specific ways, as outlined in the UML (Unified Modeling Language) diagrams that will be presented in this chapter.

### 3.2.1 Entity Objects

Entity objects represent the data and the associated behaviors in the system. In CommuniView, the entity objects include User, Business, Review, Special Offer, Trending Tag, Bookmark, and Admin. Each entity object has a set of attributes and behaviors that define its role in the system.

#### 3.2.1.1 User

A User is a distinct individual who interacts with the CommuniView web app. They utilize the platform to search for local businesses, leave reviews and ratings, and bookmark their favorite businesses. Each User is uniquely identified by a username or email address. They have attributes such as name, email, password, and a list of bookmarked businesses. Users have the ability to register, log in, search for businesses, and manage their account settings.

#### Business

A Business represents a local business entity that is listed on the CommuniView web app. Each Business is uniquely identified by a business ID. They have attributes such as name, address, phone number, category, description, business hours, photos, reviews, ratings, special offers, and trending tags. Businesses can be interacted with by Users in various ways such as searching, reviewing, rating, and bookmarking. Business owners have the ability to claim, update, and manage their Business entities.

#### BusinessClaim

A BusinessClaim represents a claim made by a user to own a business listed on the CommuniView web app. Each BusinessClaim is uniquely identified by a claim ID. They have attributes such as businessID (the ID of the business being claimed), ownerID (the ID of the user claiming the business), claimStatus (to track the status of the claim), and proofDocuments (to store any documents provided as proof of ownership). BusinessClaims can be created, edited, and deleted by users, and are processed by admins.

#### Review

A Review is a user-generated content that provides a review and rating for a local business on the CommuniView web app. Each Review is uniquely identified by a review ID. They have attributes such as author (User), content, rating value, and associated Business entity. Reviews can be created, edited, and deleted by Users, and are displayed on the associated Business Listing.

#### Special Offer

A Special Offer represents a promotional offer provided by a local business on the CommuniView web app. Each Special Offer is uniquely identified by an offer ID. They have attributes such as title, description, expiration date, and associated Business entity. Special Offers can be created, edited, and deleted by business owners, and are displayed on the associated Business Listing.

#### Trending Tag

A Trending Tag is a popular keyword or phrase associated with local businesses on the CommuniView web app. Each Trending Tag is uniquely identified by a tag ID. They have attributes such as name, popularity score, and a list of associated Business entities. Trending Tags can be created, edited, and deleted by the system or admins, and are displayed to Users for quick access to popular business categories.

#### Bookmark

A Bookmark is a saved reference to a local business on the CommuniView web app. Each Bookmark is uniquely identified by a bookmark ID. They have attributes such as associated User and Business entities. Bookmarks can be created, edited, and deleted by Users, and are stored in the User's account for easy access to their favorite businesses.

#### Admin

An Admin is an individual who manages and monitors the CommuniView web app. Each Admin is uniquely identified by a username or email address. They have attributes such as name, email, and password. Admins have the ability to log in, access the Platform Management Dashboard, and manage content and user concerns.

### 3.2.2 Boundary Objects

Boundary objects serve as the interface between the system and its actors. They facilitate the interaction between users, business owners, admins, and the system. In CommuniView, the boundary objects include Registration Form, Login Form, Search Bar, Business Listing, Review Form, Business Claim Form, Business Update Form, and Platform Management Dashboard. Each boundary object provides a specific interface for the actors to interact with the system.

#### Registration Form

The Registration Form is a boundary object that serves as an interface for user registration. It contains fields for specifying all attributes of a user, such as name, email, and password, and a control for submitting the completed form.

#### Login Form

The Login Form serves as an interface for user login. It contains fields for specifying the user's credentials such as username/email and password and a control for submitting the completed form.

#### Search Bar

The Search Bar is a boundary object that serves as an interface for business search. It allows users to input search queries, which are then processed to retrieve and display matching Business entities.

#### Business Listing

The Business Listing is a boundary object that serves as an interface for displaying detailed information about a local business. It includes various business attributes and allows users to interact with the business in various ways.

#### Review Form

The Review Form is a boundary object that serves as an interface for user-generated reviews and ratings. It contains fields for specifying the content of the review and a rating value, as well as a control for submitting the completed form.

#### Business Claim Form

The Business Claim Form is a boundary object that serves as an interface for business claiming. It contains fields for specifying the owner's identity and business ownership proof, and a control for submitting the completed form.

#### Business Update Form

The Business Update Form is a boundary object that serves as an interface for updated business information. It contains fields for specifying all attributes of a business, and a control for submitting the completed form.

#### Platform Management Dashboard

The Platform Management Dashboard is a boundary object that serves as an interface for platform monitoring and management. It allows the admin to monitor user activity, manage content, and address user concerns.

### 3.2.3 Control Objects

Control objects manage the flow of events in the system. They coordinate the interaction between entity objects and boundary objects, and control the execution of specific functionalities. In CommuniView, the control objects include User Registration Controller, User Authentication Controller, Search Controller, Trending Tags Controller, Special Offers Controller, Review and Rating Controller, Bookmark Controller, Business Claim Controller, Business Update Controller, Platform Management Controller, and APIConnector. Each control object manages a specific functionality of the system.

#### User Registration Controller

The User Registration Controller manages the user registration function. It creates a Registration Form, collects the user's information, creates a User entity, and confirms the successful registration.

#### User Authentication Controller

Manages the user authentication function. It creates a Login Form, collects the user's credentials, verifies them against existing User and Admin entities, and confirms successful authentication.

#### Search Controller

The Search Controller manages the business search function. It processes user queries, retrieves matching Business entities, and presents the results to the user.

#### Trending Tags Controller

The Trending Tags Controller manages the display of businesses associated with a specific trending tag. It retrieves the associated Business entities and presents them to the user.

#### Special Offers Controller

The Special Offers Controller manages the display of special offers. It retrieves the Special Offer entities and presents them to the user.

#### Review and Rating Controller

The Review and Rating Controller manages the review and rating function. It creates a Review Form, collects the user's information, creates a Review and Rating entity, updates the associated Business entity, and confirms the successful submission.

#### Bookmark Controller

The Bookmark Controller manages the bookmarking function. It creates a Bookmark entity, associates it with the User and Business entities, and confirms the successful bookmarking.

#### Business Claim Controller

The Business Claim Controller manages the business claiming function. It creates a BusinessClaim entity, collects the owner's information, verifies the owner's identity, updates the claimStatus attribute of the BusinessClaim entity, and confirms the successful claim.

#### 3.2.3.9 Business Update Controller

The Business Update Controller manages the business update function. It creates a Business Update Form, collects the owner's information, updates the Business entity, and confirms the successful update.

#### Platform Management Controller

The Platform Management Controller manages the platform monitoring and management functions. It presents the Platform Management Dashboard to the admin, enables them to execute management tasks, and confirms the successful execution of tasks

#### APIConnector

The APIConnector manages the interaction between the CommuniView web app and external APIs. It sends requests to and receives responses from external APIs, and processes the data for use within the system

### Data Access Layer Objects

Data access layer objects are crucial in the system architecture, serving as routes that connect user actions on the front end to the functionalities designed in the control objects on the back end. Each control object in CommuniView is paired with a corresponding data access layer object, ensuring a seamless interaction between the system and the database. These objects provide methods for creating, reading, updating, and deleting data in the database. The data access layer objects in CommuniView include UserDAO, BusinessDAO, ReviewDAO, BookmarkDAO, BusinessClaimDAO, AdminDAO, SearchDAO, TrendingTagsDAO, SpecialOffersDAO, and APIConnectorDAO.

#### 3.2.4.1 UserRegistrationDAO

Paired with the User Registration Controller, UserRegistrationDAO handles the database interactions related to the User entity. It provides methods for creating, reading, updating, and deleting User data in the database.

#### 3.2.4.2 UserAuthenticationDAO

Paired with the User Authentication Controller, UserAuthenticationDAO handles the database interactions related to user authentication. It provides methods for verifying user credentials against existing User and Admin entities in the database.

#### 3.2.4.3 SearchDAO

Paired with the Search Controller, SearchDAO handles the database interactions related to the search functionality. It provides methods for creating, reading, updating, and deleting search data in the database.

#### 3.2.4.4 TrendingTagsDAO

Paired with the Trending Tags Controller, TrendingTagsDAO handles the database interactions related to the trending tags functionality. It provides methods for creating, reading, updating, and deleting trending tags data in the database.

#### 3.2.4.5 SpecialOffersDAO

Paired with the Special Offers Controller, SpecialOffersDAO handles the database interactions related to the special offers functionality. It provides methods for creating, reading, updating, and deleting special offers data in the database.

#### **3.2.4.6** ReviewDAO

Paired with the Review and Rating Controller, ReviewDAO handles the database interactions related to the Review entity. It provides methods for creating, reading, updating, and deleting Review data in the database.

#### 3.2.4.7 BookmarkDAO

Paired with the Bookmark Controller, BookmarkDAO handles the database interactions related to the Bookmark entity. It provides methods for creating, reading, updating, and deleting Bookmark data in the database.

#### 3.2.4.8 BusinessClaimDAO

Paired with the Business Claim Controller, BusinessClaimDAO handles the database interactions related to the BusinessClaim entity. It provides methods for creating, reading, updating, and deleting BusinessClaim data in the database.

#### 3.2.4.9 BusinessUpdateDAO

Paired with the Business Update Controller, BusinessDAO handles the database interactions related to the Business entity. It provides methods for creating, reading, updating, and deleting Business data in the database.

#### 3.2.4.10 PlatformManagementDAO

Paired with the Platform Management Controller, PlatformManagementDAO handles the database interactions related to platform monitoring and management. It provides methods for presenting the Platform Management Dashboard to the admin, enabling them to execute management tasks, and confirming the successful execution of tasks in the database.

The associations among these objects are outlined in Table 3.1, which provides a clear understanding of how different objects interact with each other.

Table 3.1: Associations among CommuniView objects.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Object 1** | **Role 1** | **Multiplicity 1** | **Association** | **Object 2** | **Role 2** | **Multiplicity 2** |
| User | Bookmarks | 1..\* |  | Business | Has | 0..\* |
| User | Writes | 1..\* |  | Review | Has | 1 |
| Business | Has reviews | 1..\* |  | Review | Associated with | 1 |
| Business | Has special offers | 1..\* |  | SpecialOffer | Associated with | 1 |
| Business | Has bookmarks | 1..\* |  | Bookmark | Associated with | 1 |
| Business | Has claims | 1..\* |  | BusinessClaim | Associated with | 1 |

A screenshot of a computer

Description automatically generated

Figure 3.4: Tentative UML activity diagram for CommuniView.

## 3.3. Design

The design of the CommuniView system is guided by the goal of usability. The system is designed to be user-friendly and easy to navigate, providing an intuitive user interface, clear information architecture, and smooth user interactions. This design goal is reflected in the five subsystems of the overall system: User Management, Business Search and Discovery, Business Interaction, Business Management, and Admin Management. Each subsystem is designed to cater to specific functionalities and user interactions, contributing to the overall usability of the system.

### 3.3.1 User Management Subsystem

The User Management Subsystem is responsible for managing user registration and authentication. It includes control objects User Registration Controller and User Authentication Controller, entity object User, boundary objects Registration Form and Login Form, and data access layer objects UserRegistrationDAO and UserAuthenticationDAO.

### 3.3.2 Business Search and Discovery Subsystem

The Business Search and Discovery Subsystem is responsible for managing business search, trending tags exploration, special offers discovery, and API interactions. It includes control objects Search Controller, Trending Tags Controller, Special Offers Controller, and APIConnector, entity objects Business, Trending Tag, and Special Offer, boundary object Search Bar, and data access layer objects SearchDAO, TrendingTagsDAO, and SpecialOffersDAO.

### 3.3.3 Business Interaction Subsystem

The Business Interaction Subsystem is responsible for managing user reviews and ratings, and bookmarking of businesses. It includes control objects Review and Rating Controller and Bookmark Controller, entity objects Review and Bookmark, boundary objects Review Form and Business Listing, and data access layer objects ReviewDAO and BookmarkDAO.

### 3.3.3 Business Management Subsystem

The Business Management Subsystem is responsible for managing business claiming, business information updates, and special offers. It includes control objects Business Claim Controller and Business Update Controller, entity objects Business, BusinessClaim, and SpecialOffer, boundary objects Business Claim Form and Business Update Form, and data access layer objects BusinessUpdateDAO and BusinessClaimDAO.

### 3.3.4 Admin Management Subsystem

The Admin Management Subsystem is responsible for managing platform monitoring and management. It includes control object Platform Management Controller, entity object Admin, boundary object Platform Management Dashboard, and data access layer object PlatformManagementDAO.

The UML component diagram in Figure 3.5 provides a visual representation of these subsystems and how they interact with each other.

A screenshot of a computer

Description automatically generated

Figure 3.5: UML component diagram for CommuniView.

## 3.4 Technology Stack

The technology stack of a software system is a crucial aspect of its implementation. It determines how the system is built and how it operates. For the CommuniView system, we have chosen a technology stack that aligns with our project requirements and design considerations. This stack includes a combination of back-end technologies, front-end technologies, database technology, a server technology, and a testing technology.

### 3.4.1 Back-end Technologies

The back-end of the CommuniView system is built using Flask, a lightweight Python web framework known for its simplicity and flexibility. Flask allows us to add complexity as needed, ensuring that our system remains scalable and maintainable. We use SQLAlchemy as our ORM (Object-Relational Mapping) system. SQLAlchemy allows us to interact with our database using Python classes instead of writing SQL (Structured Query Language) queries. This makes our code cleaner, easier to understand, and more maintainable. SQLAlchemy is also database agnostic, meaning it can work with many different types of databases, providing us with flexibility for future growth [28]. Flask-Migrate is used to handle SQLAlchemy database migrations. It allows us to update our existing database schema without losing any data, ensuring that our application remains robust and reliable [29]. Flask Blueprints help us organize our application into distinct components. Each Blueprint can have its own routes, static files, and templates, making it an excellent tool for modularizing our application. This leads to more manageable and maintainable code [30]. We use Flask-WTF for form handling and validation [31], Flask-Login for user session management [32], Flask-Bcrypt for password hashing [33], Flask-Mail for email sending [34], Flask-Admin for administrative interfaces [35], Flask-RESTful for building a REST API [36], and Flask-CORS for handling Cross-Origin Resource Sharing (CORS) issues [37].

### 3.4.2 Front-end Technologies

The front-end of the CommuniView system is largely built using Javascript and Vue.js. Vue.js is a progressive Javascript framework that is easy to understand and integrate into projects. It allows us to build a user-friendly and interactive front-end for our application [38]. We use Axios for making HTTP requests from our front-end to our back-end. Axios provides a higher-level API for making HTTP requests and includes features such as automatic transformation of JSON data, progress bar support, and request cancellation [39].

### 3.4.3 Database Technology

We use SQLite as our database technology. SQLite is a simple, file-based database that is easy to use and doesn't require a separate server process. It is a good choice for small applications or for learning purposes [40]. However, if our application grows, a switch to a more robust database system may be necessary.

### 3.4.5 Testing Technology

We use Pytest for testing our application. Pytest is a Python testing framework that allows us to write simple, scalable tests for our application. It helps us ensure that our application works as expected and makes it easier to identify and fix bugs [42].