CHÚ HỀ LÀM PHẦN MỀM

LOGBLOCK SOCIAL MEDIA SERVICE
Software Architecture Document

Version 1.3

Revision History

Date	Version	Description	Author
26/11/2024	1.0	Initial documentation and Introduction of the Software Architecture Document	Ngũ Kiệt Hùng
27/11/2024	1.1	Logical View, Subsystem: Front-end View	Trình Cao An, Ngũ Kiệt Hùng
28/11/2024	1.2	Completion of Back-end Model Sub-system and Back-end Controller Sub-System	Trần Thanh Long
29/11/2024	1.3	Ngũ Kiệt Hùng: Overall system diagrams at C1, C2 and C3 level. Nguyễn Thế Thanh Long: C4: code-level diagrams for Back-end subsystem and front-end subsystem.	Ngũ Kiệt Hùng, Nguyễn Thế Thanh Long

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This Software Architecture document (**SAD**) provides a high level architecture of the System on which LogBlock will operate. The Document focuses on conveying methods of message transmission and the responsibilities as well as the dependencies between major components of the system. The System will be modeled based on the Model-View-Controller pattern.

1.2 Scope

The scope of this document will focus on providing a comprehensive view of implementation-mapped components and sub-components. This document will reference heavily to LogBlock's Vision Document, LogBlock's Use Case Specification Document and materials which are provided by Course Lecturer.

1.3 Definition, acronyms and abbreviations

- SAD: Software Architecture Document
- API: Application Programming Interface
- HTTPS: Hypertext Transfer Protocol Secure
- MVC: Model-View-Controller
- DBMS: Database Management System
- GUI: Graphical User Interface

1.4 References

- LogBlock Vision Document (version 1.3)
- LogBlock Use Case Specification Document (Version 1.4)

1.5 Overview

The structure of the document is as follow:

- The document will first define the Architectural Goals and Constraints, laid out in section 2. Architectural Goals and Constraints, where the development team outlines the key requirements and constraints that have shaped the architecture.
- The document will then provide an overview of the Use Case Diagram of the System's core functionalities in section 3. Use-Case Model.
- Section 4. Logical View will give an overview of the logical structure of the system, including its major components and dependencies among the components.
- Section 5. Deployment will provide a more in-depth description of the process of deploying the System described in Section 3 into a real machine.
- Finally, section 6. Implementation View provides a more detailed structural composition of the source tree, as well as Implementation Guide for Implementation Personnel.

2. Architectural Goals and Constraints

2.1 Goals

- The general application should be split into three major systems: The Front-end, serving GUI through the User's browser; the Back-end, processing functionalities initiated from User's Events; and the Database Management System.
- The systems run-time of major systems to be swappable during outage to maintain high uptime as required in the Vision documentation.
- Subsystems components handling functionalities should be grouped together if possible during this phase of development to avoid an explosion of file system structure.
- The system should maintain as much availability as possible, by implementing a modular design and allowing each system to be able to communicate with one or more different systems.

2.2 Constraints

- **Specialization:**

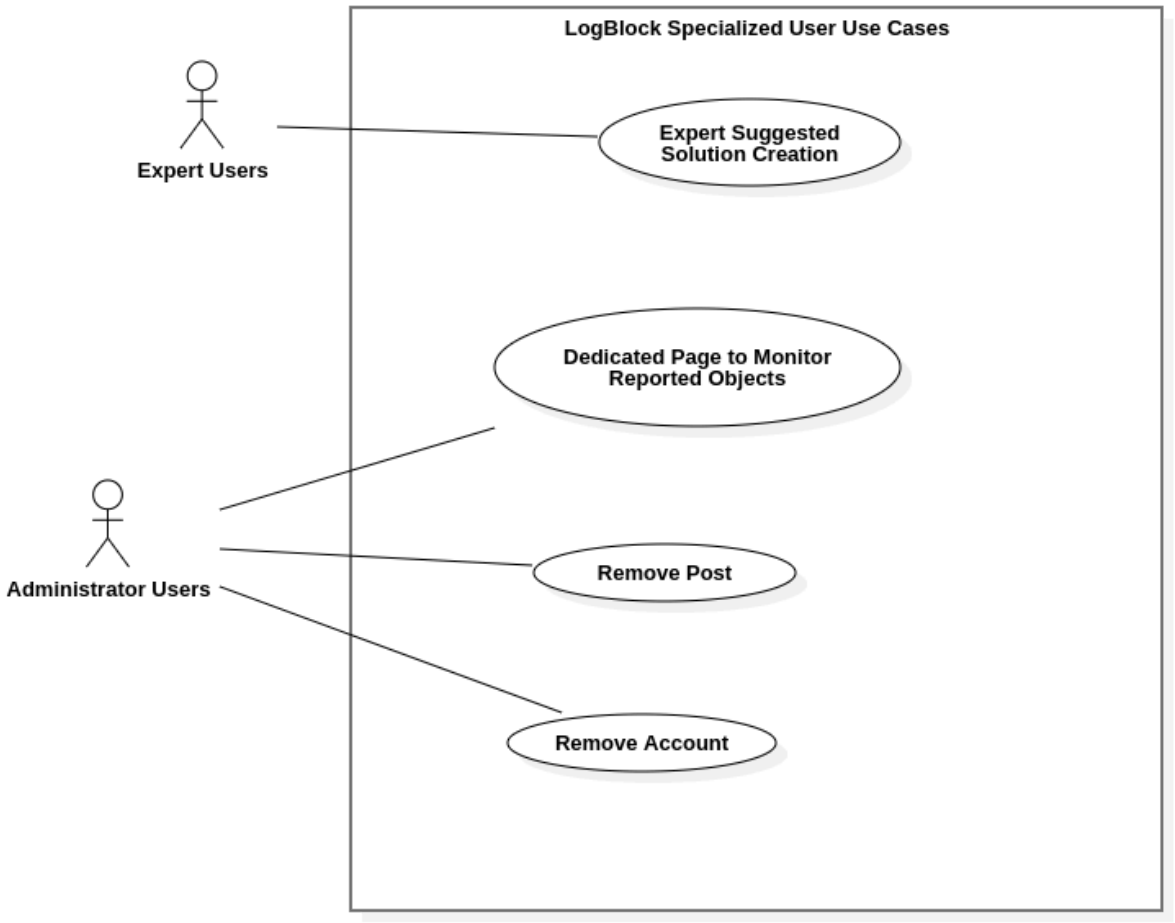
- The system which responds to the User's graphical interface request should not handle data access logics.
- Components handling GUI in the Front-end system should not directly interact with the Back-end system.

- **Technology stack:** Front-end: Next.JS (based on React.JS) framework. Back-end: Spring Boot framework. DBMS: PostgreSQL.
- **Operating Systems and Web Browsers compatibility:** The GUI serving system should be able to support multiple mainstream browsers, including Google Chrome, Microsoft Edge, Mozilla Firefox.
- **Hardware support:** The website should be lightweight to support a wide range of hardware specifications.
- **Data Encryption:** Users' information (includes personal information, password, and direct messages) should be encrypted using standard encryption protocols through cross-system communication.
- **Data Backup:** Regular data backups are performed to ensure no data is lost in case of a failure. Backup data should be compressed losslessly and should only contain vital information of the systems' current run-time state.

3. Use-Case Model

❖ Core Use Cases Diagram

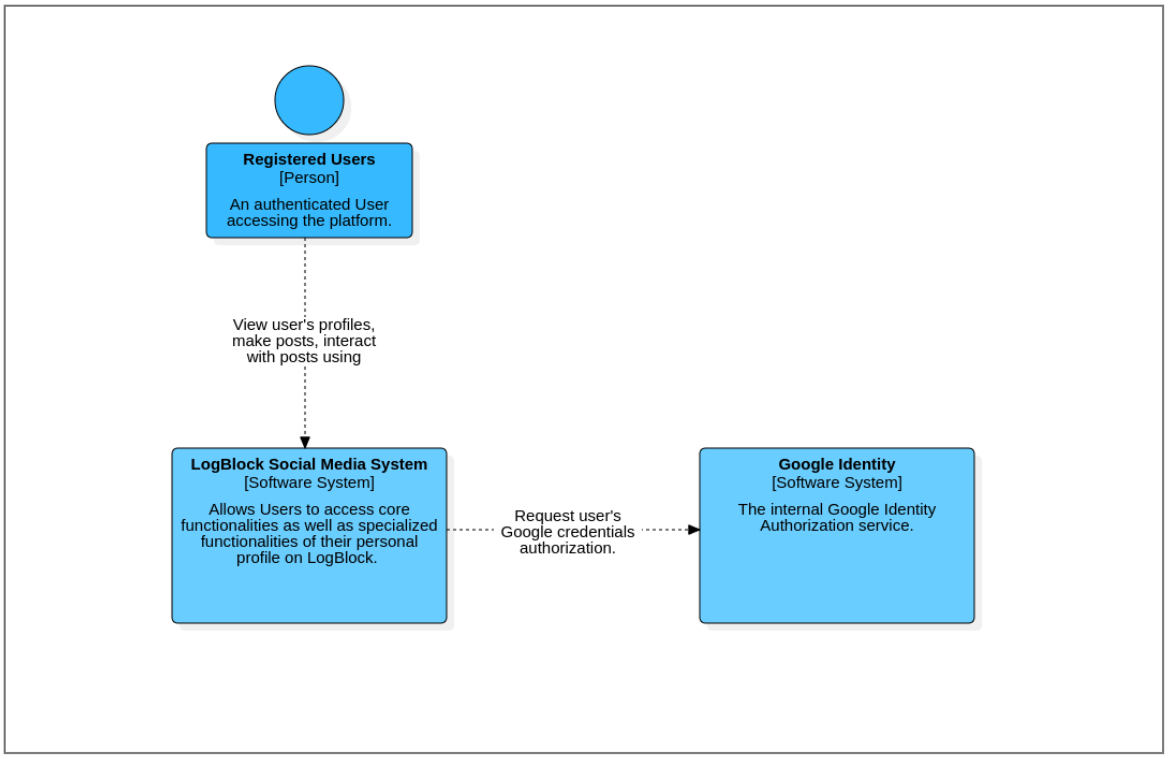
❖ **Specialized Use Cases Diagram**



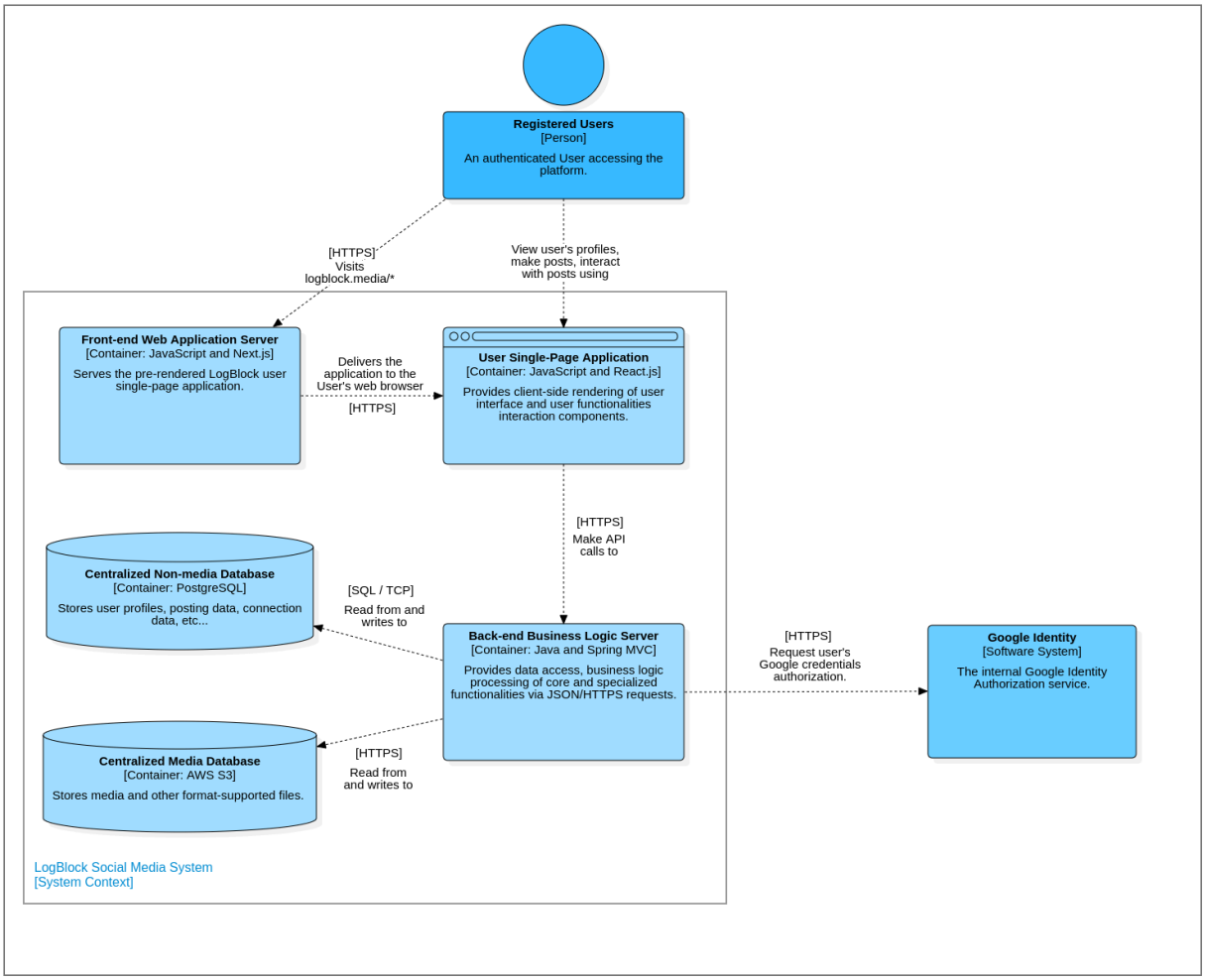
4. Logical View

LogBlock architecture follows the Model-View-Controller patterns, with primary business logic resides on the Back-End subsystem. The Front-end View subsystem primarily provides a User interface to initiate user's authenticated (and validated) requests to perform data manipulation based on the User's privilege scale.

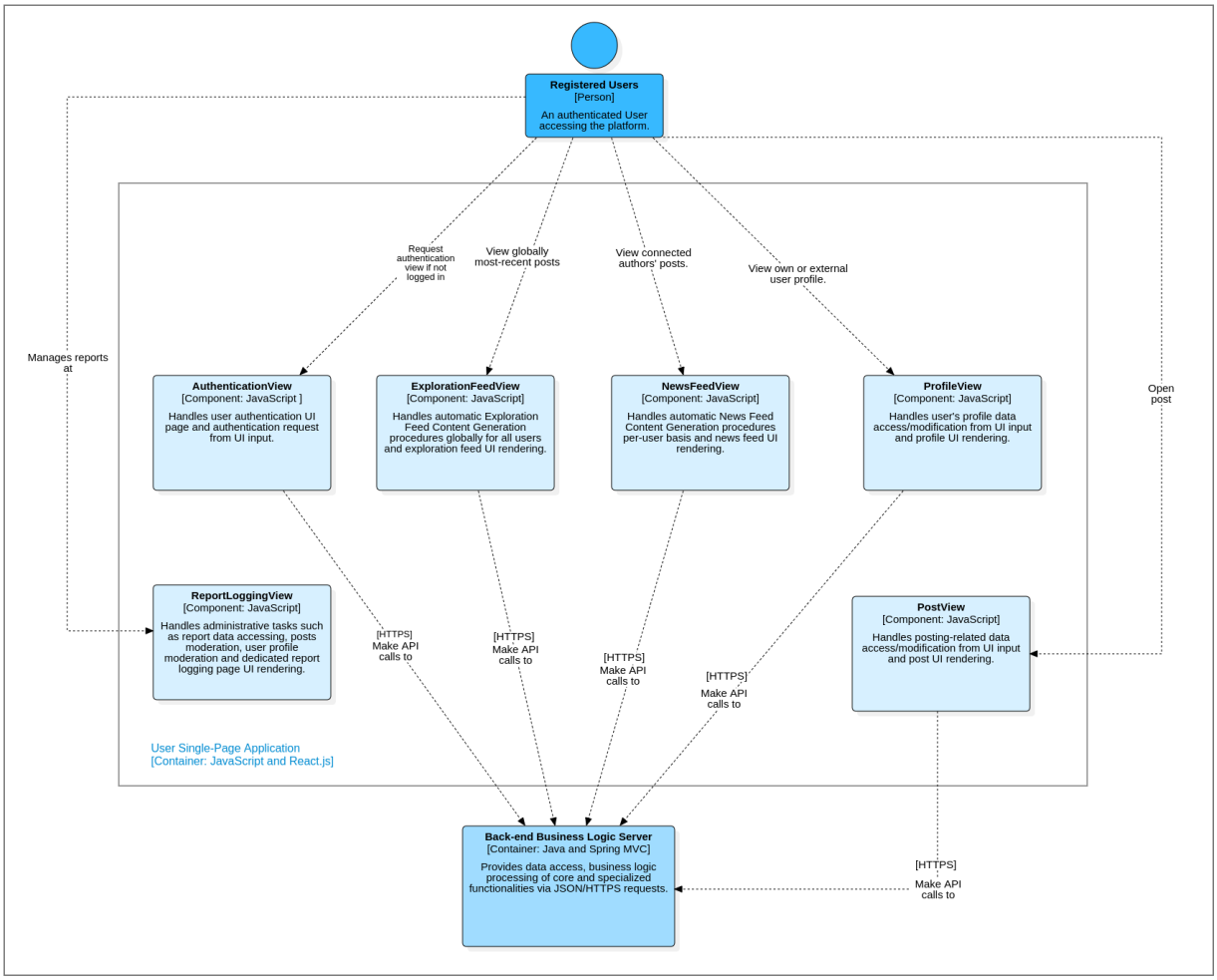
In order to present the visualization of LogBlock's architecture clearly, this document provides diagrams based on the C4 Model, comprising of C1: System Context, C2: Containers, C3.1: Front-end View Container, C3.2: Back-end Business Logic Container and C4: code-level classes of the system.



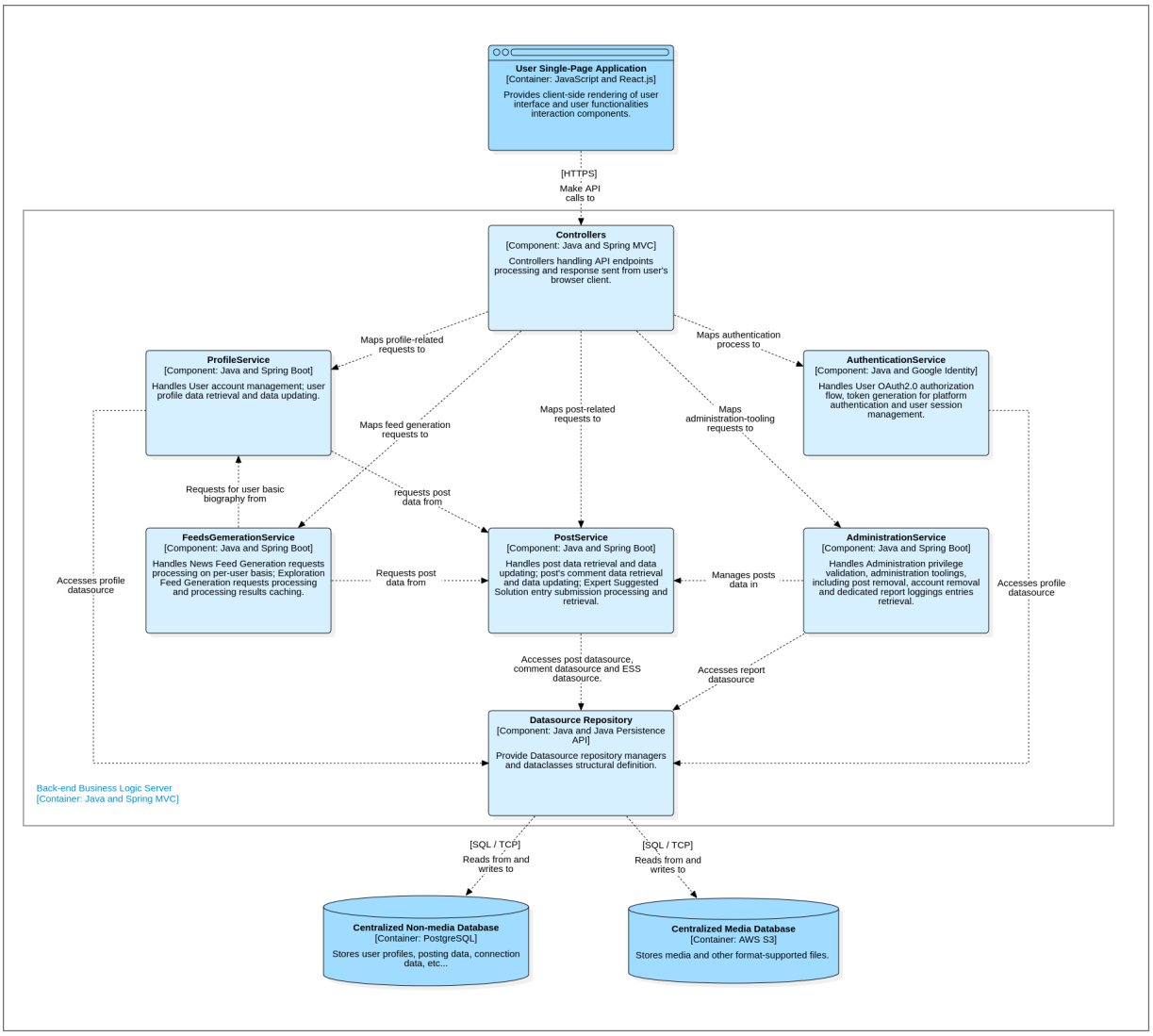
C1 level diagram: System Context



C2 level diagram: Containers



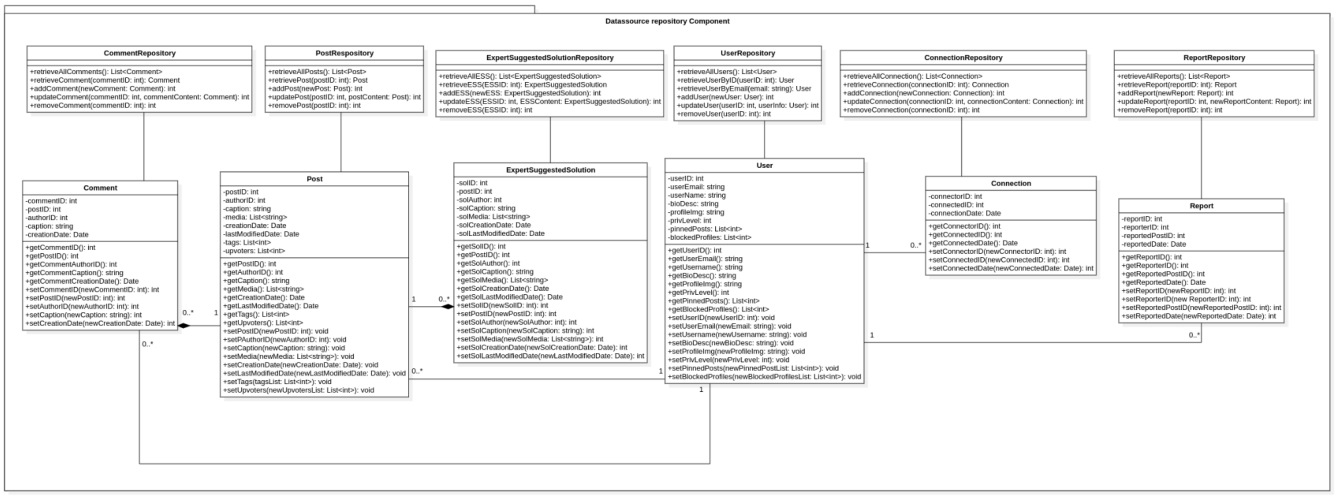
C3.1 level diagram: Front-end View Container



C3.2 level diagram: Back-end Business Logic Container

4.1 Subsystem: Back-end Business Logic

4.1.1 Component: Datasource Repository



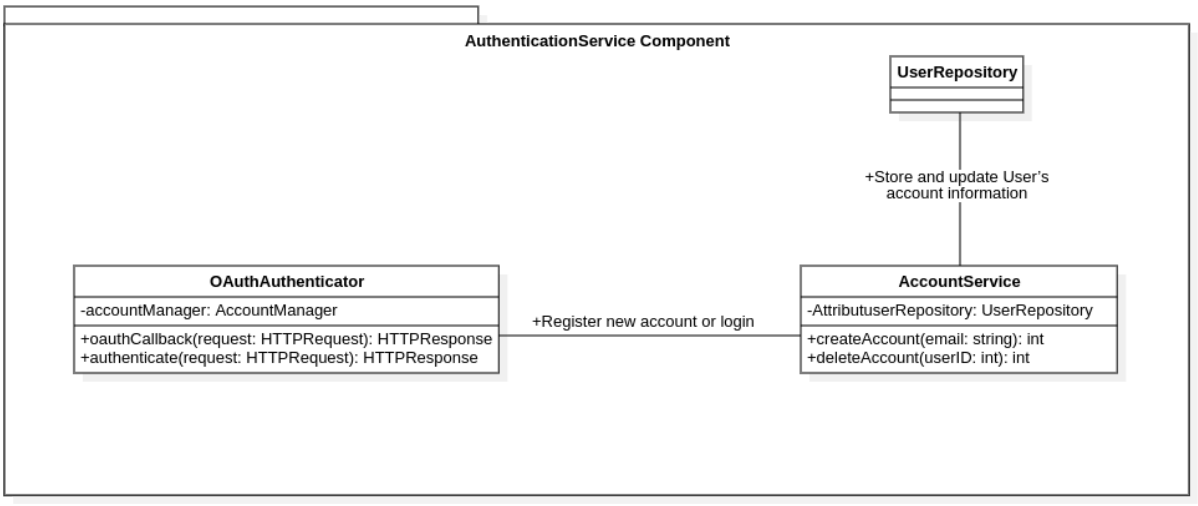
- **Responsibility:** Contains data class representing objects on the system and provides methods to retrieve and update their data to the database.
- **Class: User**
  - **Attributes:**
    - userID: int
    - userEmail: string
    - userName: string
    - bioDesc: string
    - profileImg: string
    - privLevel: int
    - pinnedPosts: List<int>
    - blockedProfiles: List<int>
    - recentlyViewedPosts: List<int>
  - **Methods:**
    - getUserID(): int
    - getUserEmail(): string
    - getUsername(): string
    - getBioDesc(): string
    - getProfileImg(): string
    - getPrivLevel(): int
    - getPinnedPosts(): List<int>
    - getRecentlyViewedPosts(): List<int>
    - getBlockedProfiles(): List<int>
    - setUserEmail(newEmail: string): void
    - setUsername(newUsername: string): void
    - setBioDesc(newBioDesc: string): void
    - setProfileImg(newProfileImg: string): void
    - setPrivLevel(newPrivLevel: int): void
    - setPinnedPosts(newPinnedPostList: List<int>): void
    - setBlockedProfiles(newBlockedProfilesList: List<int>): void
    - setRecentlyViewedPosts(newRecentlyViewPosts: List<int>): void
  - **References:** None
- **Class: UserRepository**

    - **Attributes:** None
    - **Methods:**
        - retrieveAllUsers(): List<User>
        - retrieveUserByID(userID: int): User
        - retrieveUserByEmail(email: string): User
        - addUser(newUser: User): int
        - updateUser(userID: int, userInfo: User): int
        - removeUser(userID: int): int
    - **References:** None
- **Class: Post**
    - **Attributes:**
        - postID: int
        - authorID: int
        - caption: string
        - media: List<string>
        - creationDate: Date
        - lastModifiedDate: Date
        - tags: List<int>
        - upvoters: List<int>
    - **Methods:**
        - getPostID(): int
        - getAuthorID(): int
        - getCaption(): string
        - getMedia(): List<string>
        - getCreationDate(): Date
        - getLastModifiedDate(): Date
        - getTags(): List<int>
        - getUpvoters(): List<int>
        - setPAuthorID(newAuthorID: int): void
        - setCaption(newCaption: string): void
        - setMedia(newMedia: List<string>): void
        - setCreationDate(newCreationDate: Date): void
        - setLastModifiedDate(newLastModifiedDate: Date): void
        - setTags(tagsList: List<int>): void
        - setUpvoters(newUpvotersList: List<int>): void
    - **References:** None
- **Class: PostRepository**
    - **Attributes:** None
    - **Methods:**
        - retrieveAllPosts(): List<Post>
        - retrievePost(postID: int): Post
        - addPost(newPost: Post): int
        - updatePost(postID: int, postContent: Post): int
        - removePost(postID: int): int
- **Class: Comment**
    - **Attributes:**
        - commentID: int
        - postID: int
        - authorID: int
        - caption: string
        - creationDate: Date
    - **Methods:**

        - getCommentID(): int
        - getPostID(): int
        - getCommentAuthorID(): int
        - getCommentCaption(): string
        - getCommentCreationDate(): Date
        - setAuthorID(newAuthorID: int): int
        - setCaption(newCaption: string): int
        - setCreationDate(newCreationDate: Date): int
    - **References:** None
- **Class: CommentRepository**
    - **Attributes:** None
    - **Methods:**
        - retrieveAllComments(): List<Comment>
        - retrieveComment(postID: int, commentID: int): Comment
        - addComment(postID: int, newComment: Comment): int
        - updateComment(postID: int, commentID: int, commentContent: Comment): int
        - removeComment(postID: int, commentID: int): int
    - **References:** None
- **Class: Connection**
    - **Attributes:**
        - connectorID: int
        - connectedID: int
        - connectionDate: Date
    - **Methods:**
        - getConnectorID(): int
        - getConnectedID(): int
        - getConnectedDate(): Date
        - setConnectorID(newConnectorID: int): int
        - setConnectedID(newConnectedID: int): int
        - setConnectedDate(newConnectedDate: Date): int
    - **References:** None
- **Class: ConnectionRepository**
    - **Attributes:** None
    - **Methods:**
        - retrieveAllConnection(): List<Connection>
        - retrieveConnection(connectionID: int): Connection
        - addConnection(newConnection: Connection): int
        - updateConnection(connectionID: int, connectionContent: Connection): int
        - removeConnection(connectionID: int): int
    - **References:** None
- **Class: ExpertSuggestedSolution**
    - **Attributes:**
        - solID: int
        - postID: int
        - solAuthor: int
        - solCaption: string
        - solMedia: List<string>
        - solCreationDate: Date
        - solLastModifiedDate: Date
    - **Methods:**
        - getSolID(): int
        - getPostID(): int

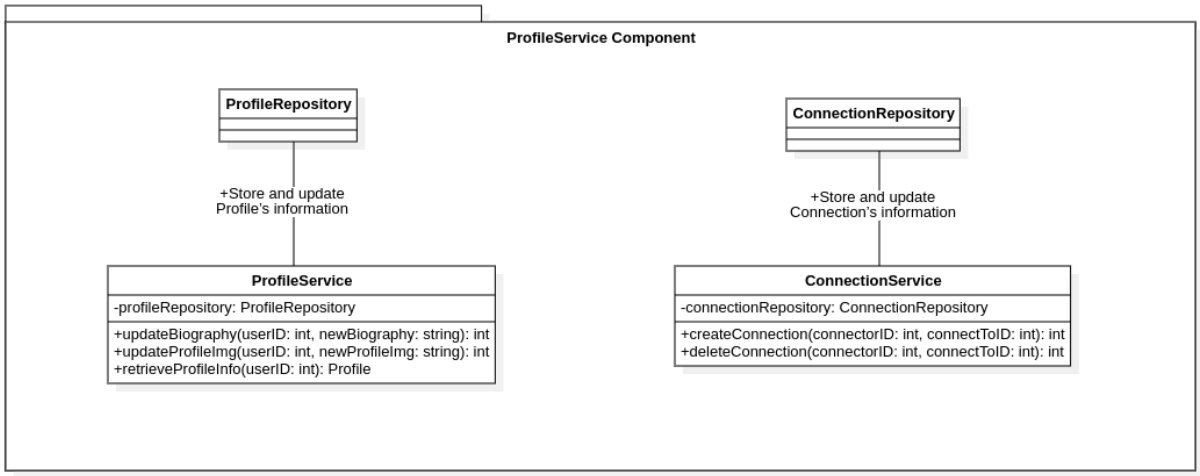
        - getSolAuthor(): int
        - getSolCaption(): string
        - getSolMedia(): List<string>
        - getSolCreationDate(): Date
        - getSolLastModifiedDate(): Date
        - setSolAuthor(newSolAuthor: int): int
        - setSolCaption(newSolCaption: string): int
        - setSolMedia(newSolMedia: List<string>): int
        - setSolCreationDate(newSolCreationDate: Date): int
        - setSolLastModifiedDate(newLastModifiedDate: Date): int
    - **References:** None
- **Class: ExpertSuggestedSolutionRepository**
    - **Attributes:** None
    - **Methods:**
        - retrieveAllESS(): List<ExpertSuggestedSolution>
        - retrieveESS(postID: int, ESSID: int): ExpertSuggestedSolution
        - addESS(newESS: ExpertSuggestedSolution): int
        - updateESS(postID: int, ESSID: int, ESSContent: ExpertSuggestedSolution): int
        - removeESS(postID: ESSID: int): int
- **Class: Report**
    - **Attribute:**
        - reportID: int
        - reporterID: int
        - reportedPostID: int
        - reportedDate: Date
    - **Methods:**
        - getReportID(): int
        - getReporterID(): int
        - getReportedPostID(): int
        - getReportedDate(): Date
        - setReporterID(new ReporterID: int): int
        - setReportedPostID(newReportedPostID: int): int
        - setReportedDate(newReportedDate: Date): int
    - **References:** None
- **Class: ReportRepository**
    - **Attributes:** None
    - **Methods:**
        - retrieveAllReports(): List<Report>
        - retrieveReport(reportID: int): Report
        - addReport(newReport: Report): int
        - updateReport(reportID: int, newReportContent: Report): int
        - removeReport(reportID: int): int
    - **References:** None

4.1.2 Component : AuthenticationService



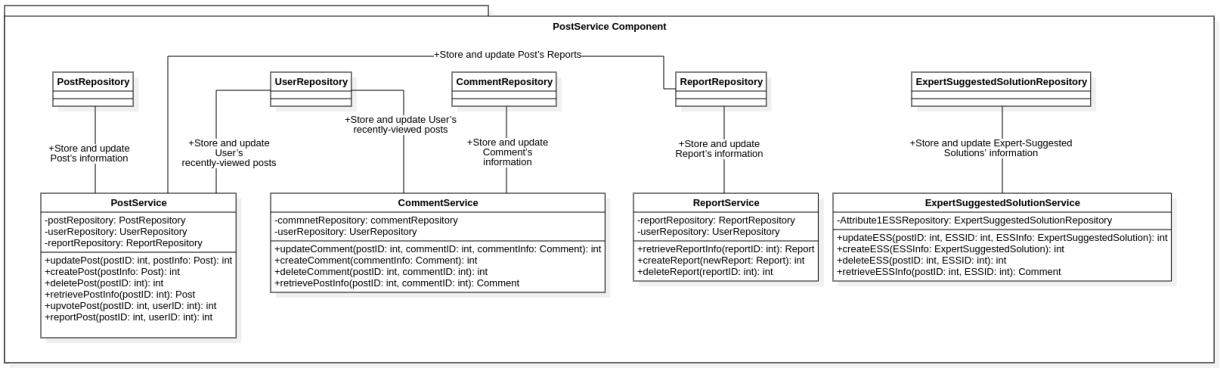
- **Responsibility:** User authentication, token generation, session management.
- **Class: OAuthAuthenticator**
  - **Attribute:** accountService: AccountService
  - **Methods:**
    - oauthCallback(request: HTTPRequest): HTTPResponse
    - authenticate(request: HTTPRequest): HTTPResponse
  - **References:**
    - **AccountService** for new account registration or login.
- **Class: AccountService**
  - **Attribute:**
    - userRepository: UserRepository
  - **Methods:**
    - createAccount(email: string): int
    - deleteAccount(userID: int): int
  - **References:**
    - **UserRepository** to store and update User's account information.

4.1.3 Component : ProfileService



- **Responsibility:** Provides methods to interact with users' profiles.
- **Class: ProfileService**
  - **Attributes:**
    - profileRepository: ProfileRepository
  - **Methods:**
    - updateBiography(userID: int, newBiography: string): int
    - updateProfileImg(userID: int, newProfileImg: string): int
    - retrieveProfileInfo(userID: int): Profile
  - **References:**
    - **ProfileRepository** to store and update Profile's information.
- **Class: ConnectionService**
  - **Attributes:**
    - connectionRepository: ConnectionRepository
  - **Methods:**
    - createConnection(connectorID: int, connectToID: int): int
    - deleteConnection(connectorID: int, connectToID: int): int
  - **References:**
    - **ConnectionRepository** to store and update Connection's information.

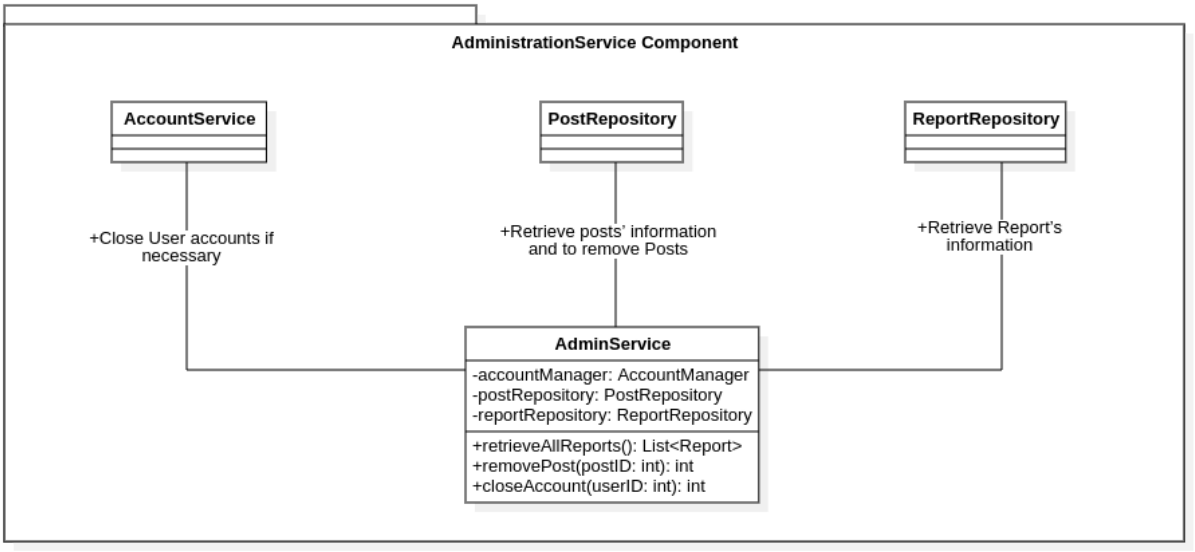
4.1.4 Component : PostService



- **Responsibility:** Provides methods to interact and manage posts.

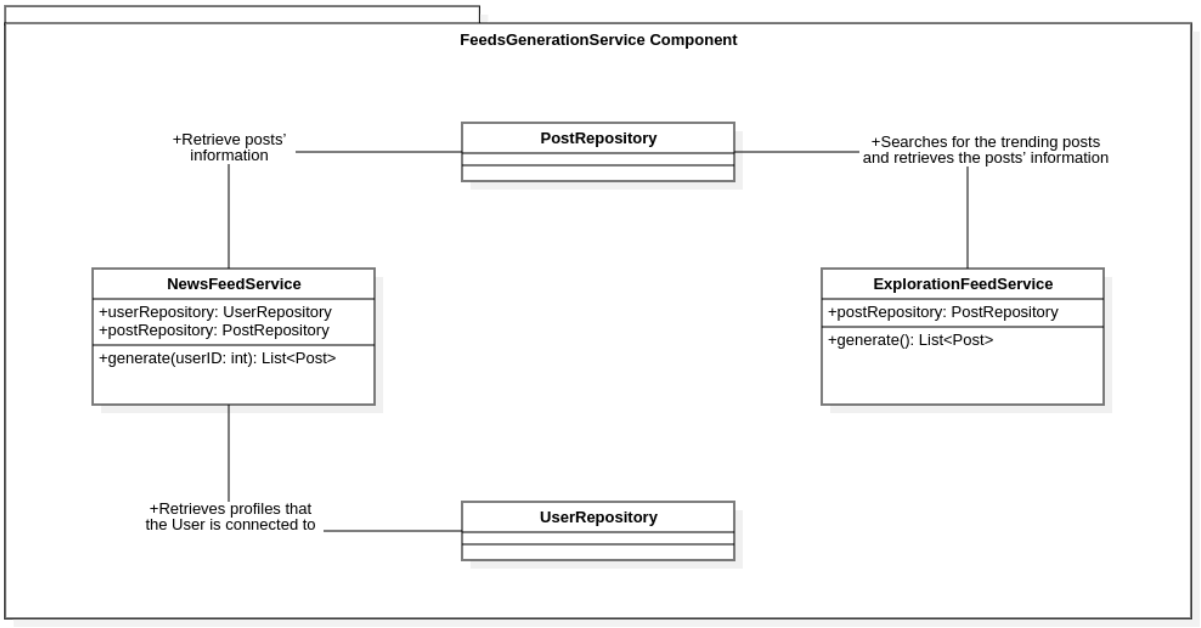
- **Class: PostService**
  - **Attributes:**
    - postRepository: PostRepository
    - userRepository: UserRepository
    - reportRepository: ReportRepository
  - **Methods:**
    - updatePost(postID: int, postInfo: Post): int
    - createPost(postInfo: Post): int
    - deletePost(postID: int): int
    - retrievePostInfo(postID: int): Post
    - upvotePost(postID: int, userID: int): int
    - reportPost(postID: int, userID: int): int
  - **References:**
    - **PostRepository** to store and update Post's information.
    - **UserRepository** to store and update User's recently-viewed posts.
    - **ReportRepository** to store and update Post's Reports.
- **Class: CommentService**
  - **Attributes:**
    - commnetRepository: commentRepository
    - userRepository: UserRepository
  - **Methods:**
    - updateComment(postID: int, commentID: int, commentInfo: Comment): int
    - createComment(commentInfo: Comment): int
    - deleteComment(postID: int, commentID: int): int
    - retrievePostInfo(postID: int, commentID: int): Comment
  - **References:**
    - **CommentRepository** to store and update Comment's information.
    - **UserRepository** to store and update User's recently-viewed posts.
- **Class: ReportService**
  - **Attributes:**
    - reportRepository: ReportRepository
    - userRepository: UserRepository
  - **Methods:**
    - retrieveReportInfo(reportID: int): Report
    - createReport(newReport: Report): int
    - deleteReport(reportID: int): int
  - **References:**
    - **ReportRepository** to store and update Report's information.
- **Class: ExpertSuggestedSolutionService**
  - **Attributes:**
    - ESSRepository: ExpertSuggestedSolutionRepository
  - **Methods:**
    - updateESS(postID: int, ESSID: int, ESSInfo: ExpertSuggestedSolution): int
    - createESS(ESSInfo: ExpertSuggestedSolution): int
    - deleteESS(postID: int, ESSID: int): int
    - retrieveESSInfo(postID: int, ESSID: int): Comment
  - **References:**
    - **ExpertSuggestedSolutionRepository** to store and update Expert-Suggested Solutions' information.

4.1.5 Component : AdministrationService



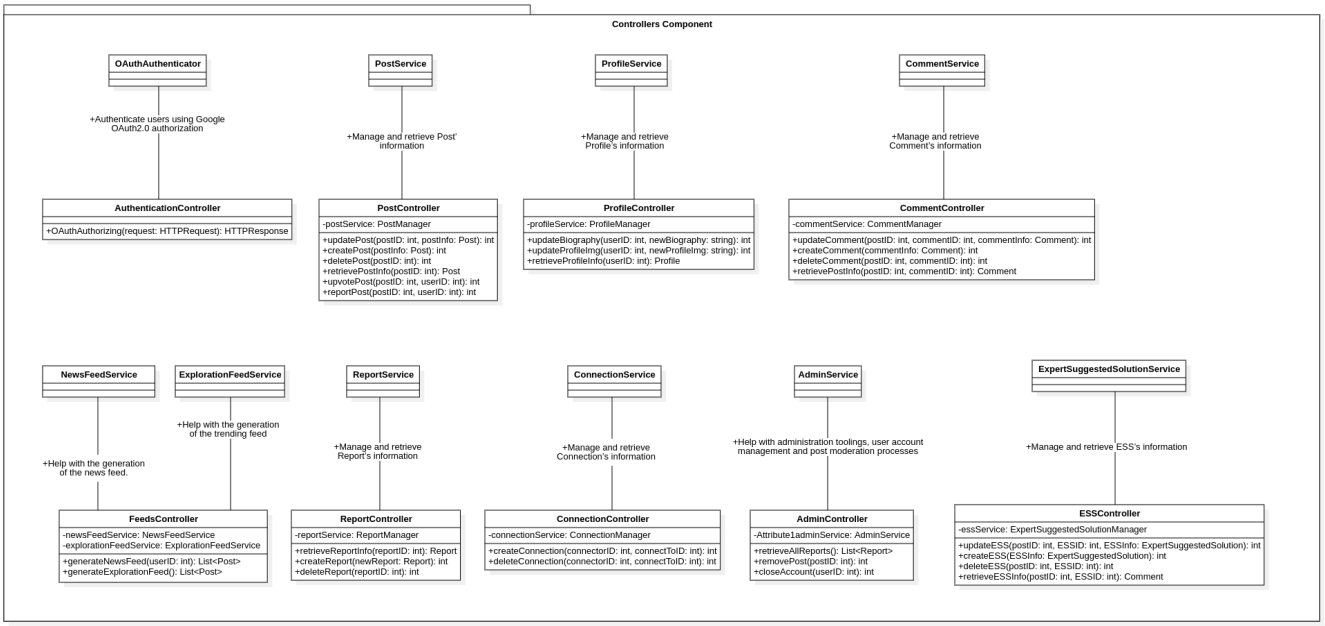
- **Responsibility:** Provides methods to help with website's administration.
- **Class: AdminService**
  - **Attributes:**
    - accountService: AccountService
    - postRepository: PostRepository
    - reportRepository: ReportRepository
  - **Methods:**
    - retrieveAllReports(): List<Report>
    - removePost(postID: int): int
    - closeAccount(userID: int): int
  - **References:**
    - **AccountService** to close User accounts if necessary.
    - **PostRepository** to retrieve posts' information and to remove Posts.
    - **ReportRepository** to retrieve Report's information.

4.1.6 Component : FeedsGenerationService



- **Responsibility:** Provides methods to generate posts on the News Feed and the Trending Feed.
- **Class: NewsFeedService**
  - **Attributes:**
    - userRepository: UserRepository
    - postRepository: PostRepository
  - **Methods:**
    - generate(userID: int): List<Post>
  - **References:**
    - **UserRepository** to retrieve profiles that the User is connected to.
    - **PostRepository** to retrieve posts' information.
- **Class: ExplorationFeedService**
  - **Attributes:**
    - postRepository: PostRepository
  - **Methods:**
    - generate(): List<Post>
  - **References:**
    - **PostRepository** to search for the trending posts and retrieve the posts' information.

4.1.7 Component: Controllers



- **Responsibility:** Provides Back-end controller to map with the Front-end's requests.
- **Class: AuthenticationController**
  - **Attribute:** None
  - **Methods:**
    - OAuthAuthorizing(request: HTTPRequest): HTTPResponse
  - **References:**
    - **OAuthAuthenticator** to authenticate users using Google OAuth2.0 authorization.
- **Class: PostController**
  - **Attribute:**
    - postService: PostService
  - **Methods:**
    - updatePost(postID: int, postInfo: Post): int
    - createPost(postInfo: Post): int
    - deletePost(postID: int): int
    - retrievePostInfo(postID: int): Post
    - upvotePost(postID: int, userID: int): int
    - reportPost(postID: int, userID: int): int
  - **References:**
    - **PostService** to manage and retrieve Post' information.
- **Class: ProfileController**
  - **Attribute:**
    - profileService: ProfileService
  - **Methods:**
    - updateBiography(userID: int, newBiography: string): int
    - updateProfileImg(userID: int, newProfileImg: string): int
    - retrieveProfileInfo(userID: int): Profile
  - **References:**
    - **ProfileService** to manage and retrieve Profile's information.

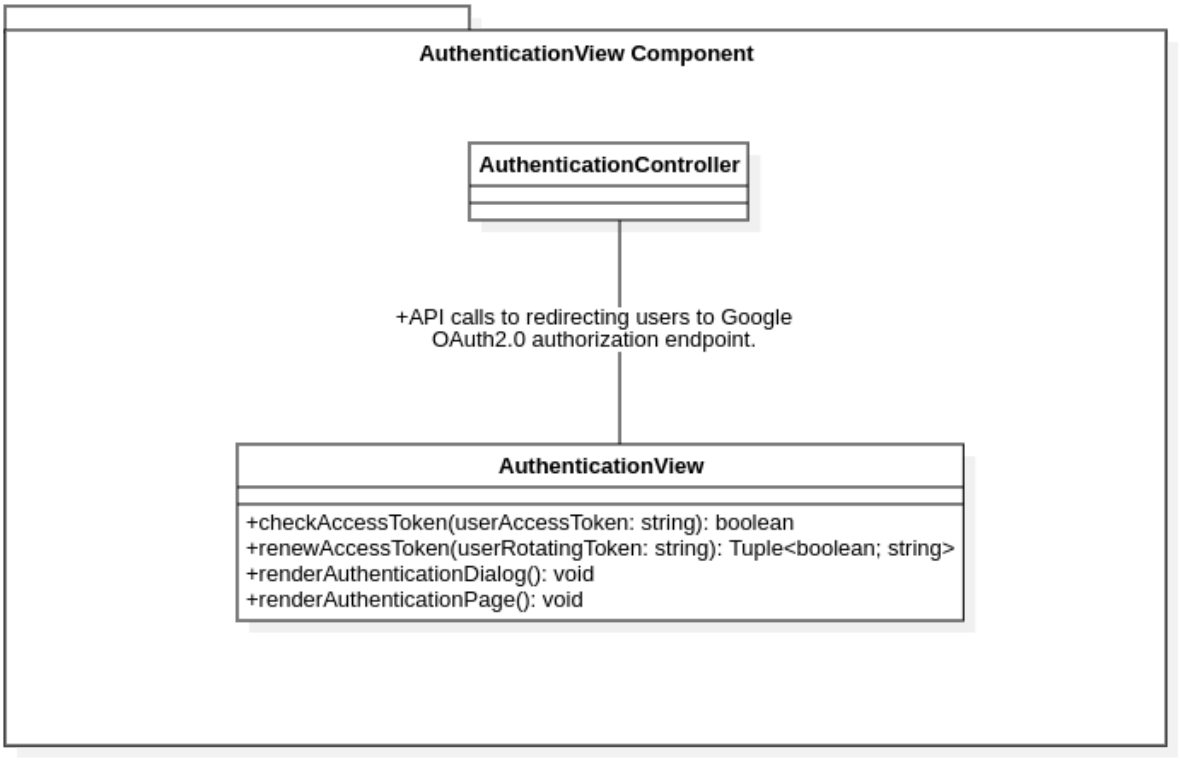
- **Class: CommentController**
  - **Attribute:**
    - commentService: CommentService
  - **Methods:**
    - updateComment(postID: int, commentID: int, commentInfo: Comment): int
    - createComment(commentInfo: Comment): int
    - deleteComment(postID: int, commentID: int): int
    - retrievePostInfo(postID: int, commentID: int): Comment
  - **References:**
    - **CommentService** to manage and retrieve Comment's information.
- **Class: ESSController**
  - **Attribute:**
    - essService: ExpertSuggestedSolutionService
  - **Methods:**
    - updateESS(postID: int, ESSID: int, ESSInfo: ExpertSuggestedSolution): int
    - createESS(ESSInfo: ExpertSuggestedSolution): int
    - deleteESS(postID: int, ESSID: int): int
    - retrieveESSInfo(postID: int, ESSID: int): Comment
  - **References:**
    - **ExpertSuggestedSolutionService** to manage and retrieve ESS's information.
- **Class: ConnectionController**
  - **Attribute:**
    - connectionService: ConnectionService
  - **Methods:**
    - createConnection(connectorID: int, connectToID: int): int
    - deleteConnection(connectorID: int, connectToID: int): int
  - **References:**
    - **ConnectionService** to manage and retrieve Connection's information.
- **Class: ReportController**
  - **Attribute:**
    - reportService: ReportService
  - **Methods:**
    - retrieveReportInfo(reportID: int): Report
    - createReport(newReport: Report): int
    - deleteReport(reportID: int): int
  - **References:**
    - **ReportService** to manage and retrieve Report's information.
- **Class: FeedsController**
  - **Attribute:**
    - newsFeedService: NewsFeedService
    - explorationFeedService: ExplorationFeedService
  - **Methods:**
    - generateNewsFeed(userID: int): List<Post>
    - generateExplorationFeed(): List<Post>
  - **References:**
    - **NewsFeedService** to help with the generation of the news feed.
    - **ExplorationFeedService** to help with the generation of the trending feed.
- **Class: AdminController**
  - **Attribute:**
    - adminService: AdminService
  - **Methods:**
    - retrieveAllReports(): List<Report>

- removePost(postID: int): int
- closeAccount(userID: int): int

- **References:**
  - **AdminService** to help with administration toolings, user account management and post moderation processes.

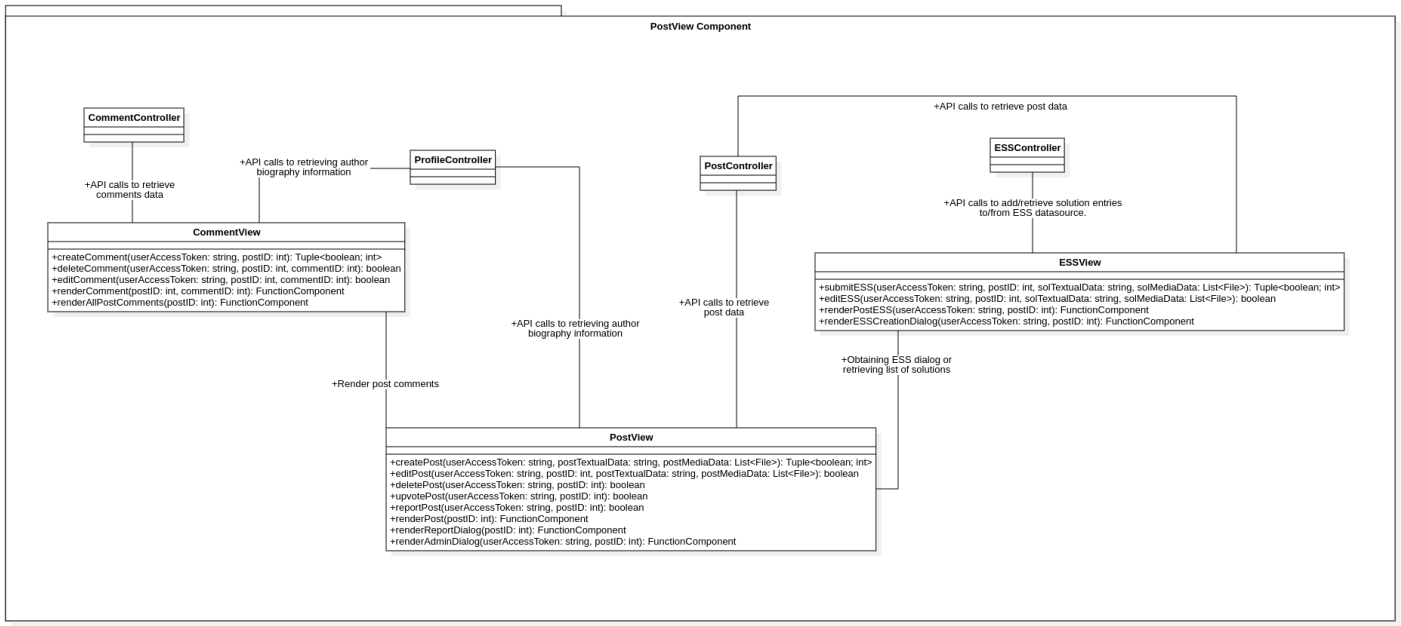
4.2 Subsystem: Front-end View

4.2.1 Component : AuthenticationView



- **Responsibility:** Provide Authentication user interface and manage user locally-stored authentication state.
- **Class:** AuthenticationView
  - **Attribute:** None
  - **Methods:**
    - checkAccessToken(userAccessToken: string): boolean
    - renewAccessToken(userRotatingToken: string): Tuple<boolean; string>
    - renderAuthenticationDialog(): void
    - renderAuthenticationPage(): void
  - **References:**
    - **AuthenticationController:** To redirect users to Back-end API endpoint for Google OAuth2.0 authorization flow.

4.2.2 Component: PostView

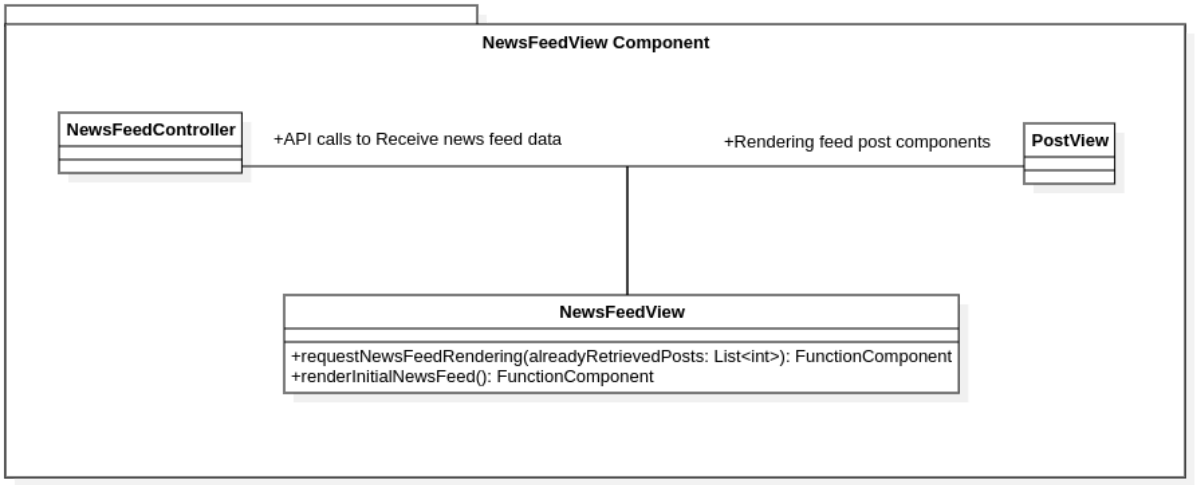


- **Responsibility:** Provides post data access API calls, renders posts interface and handles user post-interaction inputs.
- **Class:** PostView
    - **Attribute:** None
    - **Methods:**
        - createPost(userAccessToken: string, postTextualData: string, postMediaData: List<File>): Tuple<boolean; int>
        - editPost(userAccessToken: string, postID: int, postTextualData: string, postMediaData: List<File>): boolean
        - deletePost(userAccessToken: string, postID: int): boolean
        - upvotePost(userAccessToken: string, postID: int): boolean
        - reportPost(userAccessToken: string, postID: int): boolean
        - renderPost(postID: int): FunctionComponent
        - renderReportDialog(postID: int): FunctionComponent
        - renderAdminDialog(userAccessToken: string, postID: int): FunctionComponent
    - **References:**
        - **PostController:** To retrieve post data.
        - **ProfileController:** To retrieve basic post author biography information.
        - **ESSView:** To obtain Expert Suggested Solution creation dialog rendering or retrieve list of solutions of post's original author.
        - **CommentView:** To render post comments.
- **Class:** CommentView
    - **Attribute:** None
    - **Methods:**
        - createComment(userAccessToken: string, postID: int): Tuple<boolean; int>
        - deleteComment(userAccessToken: string, postID: int, commentID: int): boolean
        - editComment(userAccessToken: string, postID: int, commentID: int): boolean
        - renderComment(postID: int, commentID: int): FunctionComponent
        - renderAllPostComments(postID: int): FunctionComponent

- **References:**
  - **CommentController:** to retrieve comments data.
  - **ProfileController:** To retrieve basic comment author biography information.

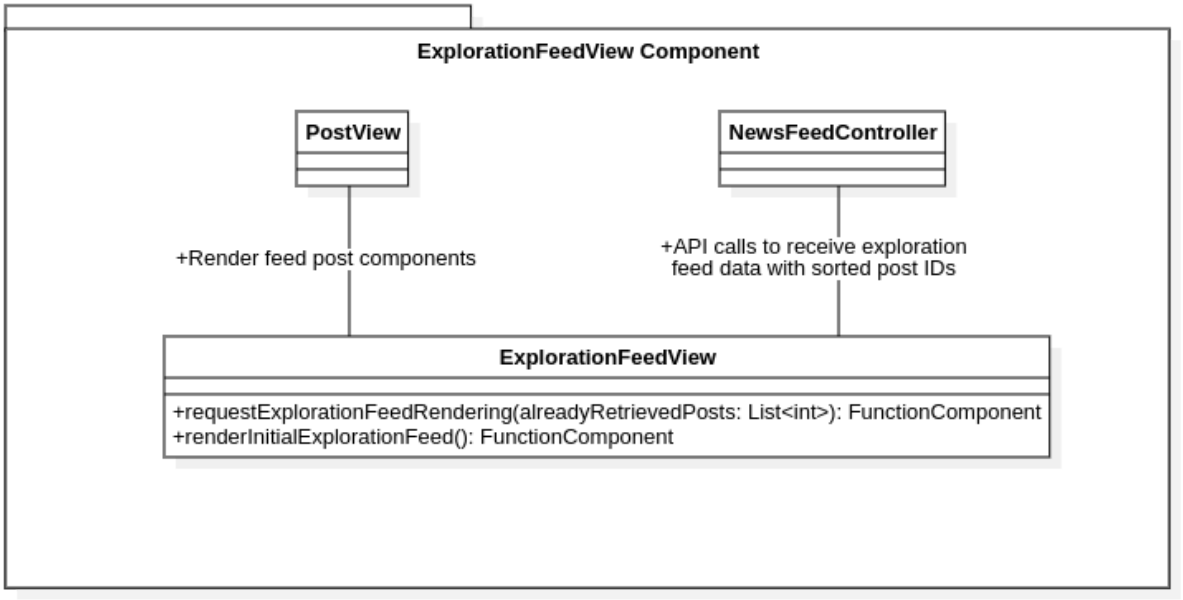
- **Class:** ESSView
  - **Attribute:** None
  - **Methods:**
    - submitESS(userAccessToken: string, postID: int, solTextualData: string, solMediaData: List<File>): Tuple<boolean; int>
    - editESS(userAccessToken: string, postID: int, solTextualData:string, solMediaData: List<File>): boolean
    - renderPostESS(userAccessToken: string, postID: int): FunctionComponent
    - renderESSCreationDialog(userAccessToken: string, postID: int): FunctionComponent
  - **References :**
    - **PostController:** Attach expert-provided solutions to posts.
    - **ESSController:** To add an entry/retrieve solution entries into/from the ESS datasource.

4.2.3 Component: NewsFeedView



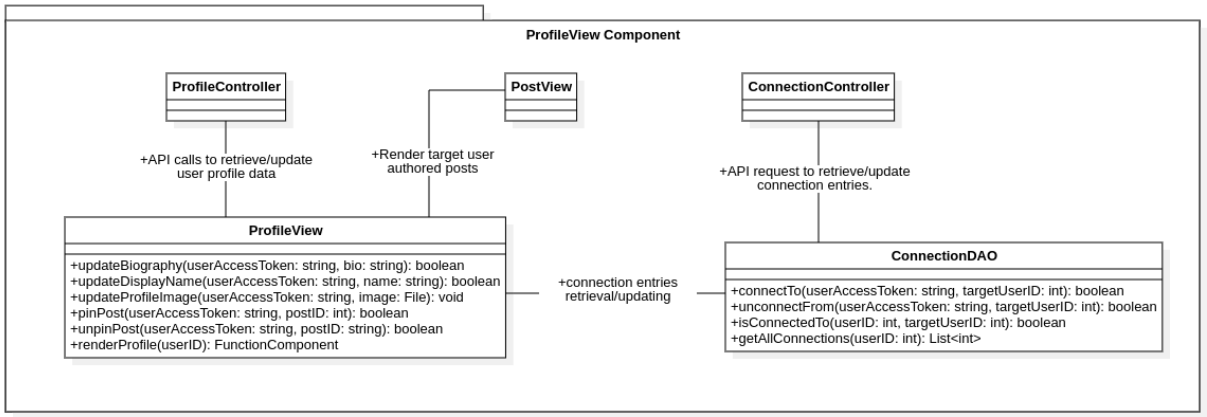
- **Responsibility:** Retrieves, generates, and renders content for the user's News Feed, based on their connections and activities.
- **Class:** NewsFeedView
  - **Attribute:** None
  - **Methods:**
    - requestNewsFeedRendering(alreadyRetrievedPosts: List<int>): FunctionComponent
    - renderInitialNewsFeed(): FunctionComponent
  - **References:**
    - **PostView:** To render feed post components.
    - **NewsFeedController:** To receive news feed data comprising sorted post ids to render for the user.

4.2.4 Component: ExplorationFeedView



- **Responsibility:** Retrieves, generates and renders global or trending posts in the Exploration Feed, independent of user connections.
- **Class:** ExplorationFeedView
  - **Attribute:** None
  - **Methods:**
    - requestExplorationFeedRendering(alreadyRetrievedPosts: List<int>): FunctionComponent
    - renderInitialExplorationFeed(): FunctionComponent
  - **References:**
    - **PostView:** To render feed post components.
    - **NewsFeedController:** To receive exploration feed data comprising sorted post ids to render for the user.

4.2.5 Component: ProfileView

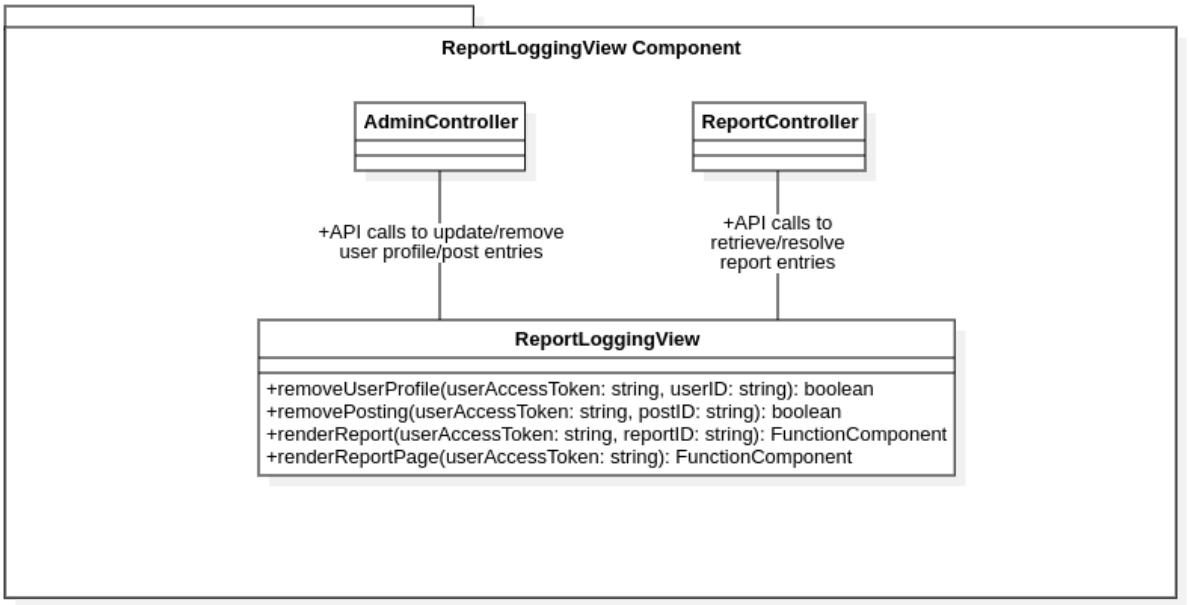


- **Responsibility:** Provides profile data access API calls, renders profile interface and handles user

profile-interaction inputs.

- **Class:** ProfileView
  - **Attribute:** None
  - **Methods:**
    - updateBiography(userAccessToken: string, bio: string): boolean
    - updateDisplayName(userAccessToken: string, name: string) : boolean
    - updateProfileImage(userAccessToken: string, image: File): void
    - pinPost(userAccessToken: string, postID: int): boolean
    - unpinPost(userAccessToken: string, postID: string): boolean
    - renderProfile(userID): FunctionComponent
  - **References :**
    - **ProfileController:** To retrieve/update user profile data.
    - **ConnectionDAO:** for connection entries retrieval/updating
    - **PostView:** To render target user authored posts.
- **Class:** ConnectionDAO
  - **Attribute:** None
  - **Methods:**
    - connectTo(userAccessToken: string, targetUserID: int): boolean
    - unconnectFrom(userAccessToken: string, targetUserID: int): boolean
    - isConnectedTo(userID: int, targetUserID: int): boolean
    - getAllConnections(userID: int): List<int>
  - **References:**
    - **ConnectionController:** To retrieve/update connection entries.

4.2.6 Component : ReportLoggingView



- **Responsibility:** Provides user interface for Administration user's dedicated report logging page and handles administrative operations inputs.
- **Class:** ReportLoggingView
  - **Methods:**
    - removeUserProfile(userAccessToken: string, userID: string): boolean
    - removePosting(userAccessToken: string, postID: string): boolean
    - renderReport(userAccessToken: string, reportID: string): FunctionComponent

- renderReportPage(userAccessToken: string): FunctionComponent

- **References :**
  - **AdminController:** To update/remove user profile/post entries.
  - **ReportController:** To retrieve/resolve report entries.

5. Deployment

6. Implementation View