Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering
Fall 2023

Deliverable 5

Class Diagram

Team 5

Cahit Ediz Civan (22003206)
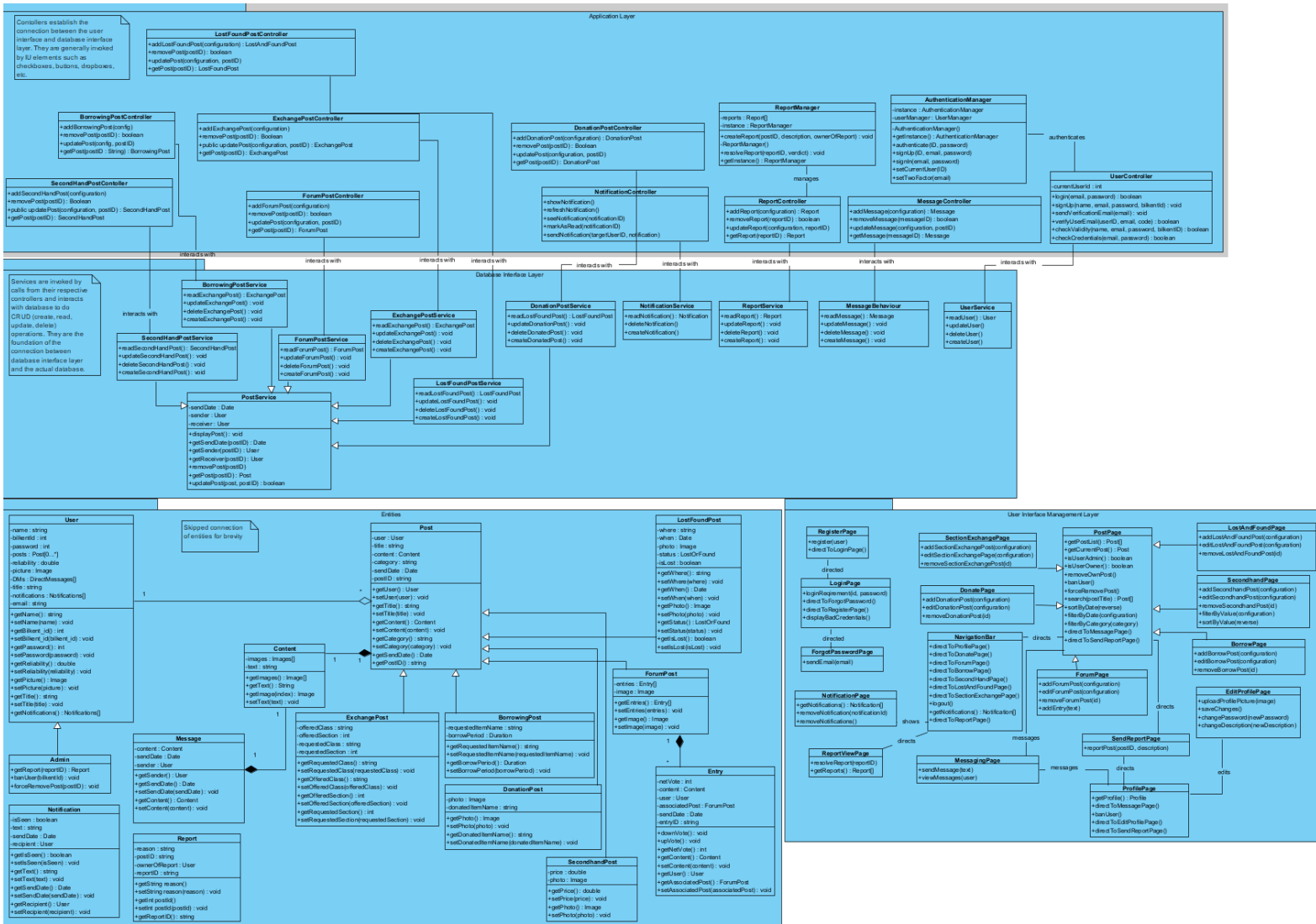Efe Kaan Fidancı (22102589)
Emir Tuğlu (22003165)
Görkem Kadir Solun (22003214)
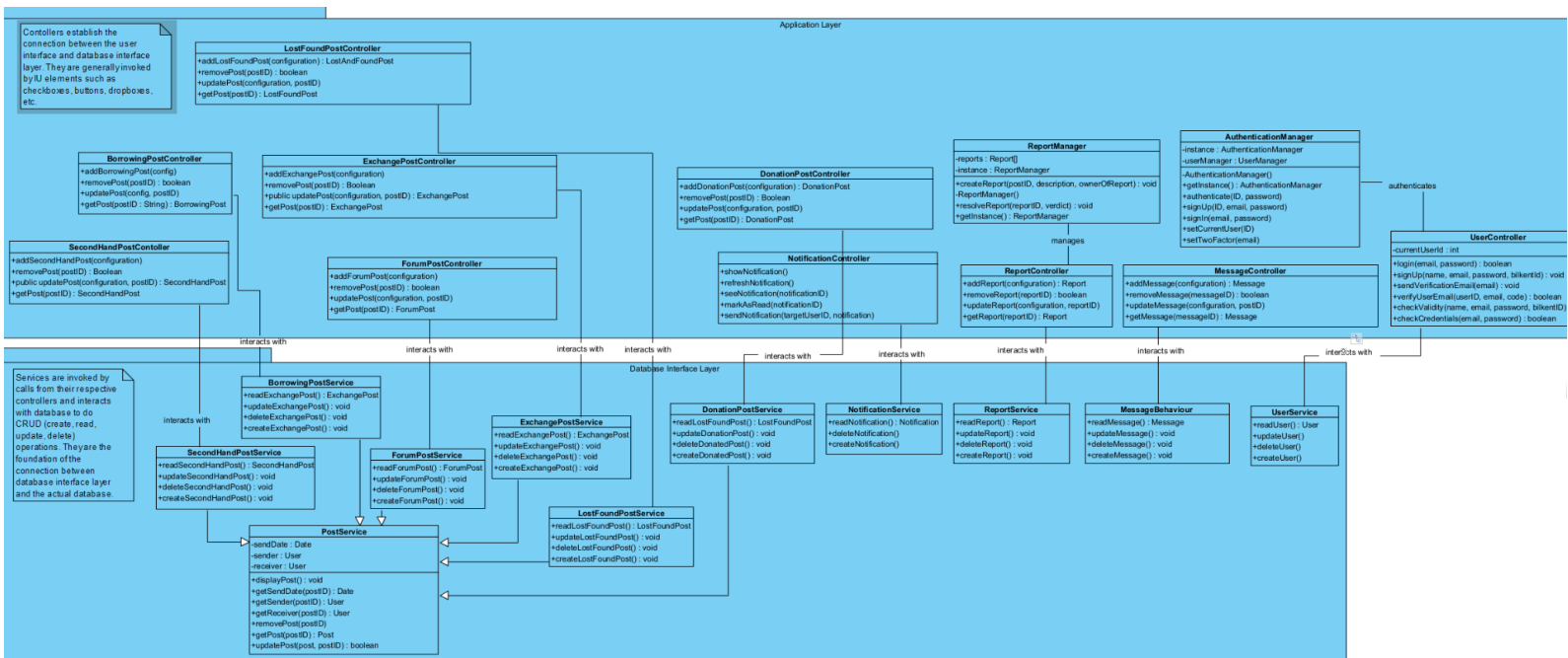Mete Enes Yılmaz (22102849)

Table of Contents

Solution Level Class Diagram

Application Layer

Controllers establish the connection between the user interface and database interface layer. They are generally invoked by UI elements such as checkboxes, buttons, dropboxes, etc.

LostFoundPostController
+addLostFoundPost(configuration) : LostAndFoundPost
+removePost(postID) : Boolean
+updatePost(configuration, postID)
+getPost(postID : String) : LostFoundPost

BorrowingPostController
+addBorrowingPost(config)
+removePost(postID) : boolean
+updatePost(config, postID)
+getPost(postID : String) : BorrowingPost

ExchangePostController
+addExchangePost(configuration)
+removePost(postID) : Boolean
+public updatePost(configuration, postID) : ExchangePost
+getPost(postID) : ExchangePost

SecondHandPostController
+addSecondHandPost(configuration)
+removePost(postID) : Boolean
+public updatePost(configuration, postID) : SecondHandPost
+getPost(postID) : SecondHandPost

ForumPostController
+addForumPost(configuration)
+removePost(postID) : boolean
+updatePost(configuration, postID)
+getPost(postID) : ForumPost

DonationPostController
+addDonationPost(configuration) : DonationPost
+removePost(postID) : Boolean
+updatePost(configuration, postID)
+getPost(postID) : DonationPost

NotificationController
+showNotification()
+refreshNotification()
+seeNotification(notificationID)
+markAsRead(notificationID)
+sendNotification(targetUserID, notification)

ReportManager
-reports : Report[]
-instance : ReportManager
+createReport(postID, description, ownerOfReport) : void
+ReportManager()
+resolveReport(reportID, verdict) : void
+getInstance() : ReportManager

ReportController
+addReport(configuration) : Report
+removeReport(reportID) : boolean
+updateReport(configuration, reportID)
+getReport(reportID) : Report

MessageController
+addMessage(configuration) : Message
+removeMessage(messageID) : boolean
+updateMessage(configuration, postID)
+getMessage(messageID) : Message

AuthenticationManager
-instance : AuthenticationManager
-userManager : UserManager
+AuthenticationManager()
+getInstance() : AuthenticationManager
+authenticate(ID, password)
+signUp(ID, email, password)
+signIn(email, password)
+setCurrentUser(ID)
+setTwoFactor(email)

UserController
-currentUserId : int
+login(email, password) : boolean
+signUp(name, email, password, bikenItId) : void
+sendVerificationEmail(email) : void
+verifyUser(userID, email, code) : boolean
+checkValidity(name, email, password, bikenID) : boolean
+checkCredentials(email, password) : boolean

authenticates

manages

Database Interface Layer

Services are invoked by calls from their respective controllers and interacts with database to do CRUD (create, read, update, delete) operations. They are the foundation of the connection between database interface layer and the actual database.

interacts with

BorrowingPostService
+readExchangePost() : ExchangePost
+updateExchangePost() : void
+deleteExchangePost() : void
+createExchangePost() : void

SecondHandPostService
+readSecondHandPost() : SecondHandPost
+updateSecondHandPost() : void
+deleteSecondHandPost() : void
+createSecondHandPost() : void

ForumPostService
+readForumPost() : ForumPost
+updateForumPost() : void
+deleteForumPost() : void
+createForumPost() : void

ExchangePostService
+readExchangePost() : ExchangePost
+updateExchangePost() : void
+deleteExchangePost() : void
+createExchangePost() : void

LostFoundPostService
+readLostFoundPost() : LostFoundPost
+updateLostFoundPost() : void
+deleteLostFoundPost() : void
+createLostFoundPost() : void

DonationPostService
+readLostFoundPost() : LostFoundPost
+updateDonationPost() : void
+deleteDonationPost() : void
+createDonatedPost() : void

NotificationService
+readNotification() : Notification
+deleteNotification()
+createNotification()

ReportService
+readReport() : Report
+updateReport() : void
+deleteReport() : void
+createReport() : void

MessageBehaviour
+readMessage() : Message
+updateMessage()
+deleteMessage() : void
+createMessage() : void

UserService
+readUser() : User
+updateUser()
+deleteUser()
+createUser()

PostService
-sendDate : Date
-sender : User
-receiver : User
+displayPost() : void
+getSendDate(postID) : Date
+getSender(postID) : User
+getReceiver(postID) : User
+removePost(postID)
+getPost(postID) : Post
+updatePost(post, postID) : boolean

Entities

Skipped connection of entities for brevity

User
-name : string
-bikenItId : int
-password : int
-posts : Post[0...*]
-reliability : double
-picture : Image
-title : string
-notifications : Notifications[]
-email : string
+getName() : string
+setName(name) : void
+getBikenId_id() : int
+setBikenId_id(bikenId_id) : void
+getPassword() : int
+setPassword(password) : void
+getReliability() : double
+setReliability(reliability) : void
+getPicture() : Image
+setPicture(picture) : void
+getTitle() : string
+setTitle(title) : void
+getNotifications() : Notifications[]

Admin
+getReport(reportID) : Report
+banUser(bikenItId) : void
+forceRemovePost(postID) : void

Notification
-isSeen : boolean
-text : string
-sendDate : Date
-recipient : User
+getIsSeen() : boolean
+setIsSeen(isSeen) : void
+getText() : string
+setText(text) : void
+getSendDate() : Date
+setSendDate(sendDate) : void
+getRecipient() : User
+setRecipient(recipient) : void

Content
-images : Images[]
-text : string
+getImages() : Image[]
+getText() : String
+getImage(index) : Image
+setText(text) : void

Message
-content : Content
-sendDate : Date
-sender : User
+getSender() : User
+getSendDate() : Date
+setSendDate(sendDate) : void
+getContent() : Content
+setContent(content) : void

Report
-reason : string
-postID : string
-ownerOfReport : User
-reportID : string
+getString reason()
+setString reason(reason) : void
+getInt postId()
+setInt postId(postId) : void
+getReportID() : string

Post
-user : User
-title : string
-content : Content
-category : string
-sendDate : Date
-postID : string
+getUser() : User
+setUser(user) : void
+getTitle() : string
+setTitle(title) : void
+getContent() : Content
+setContent(content) : void
+getCategory() : string
+setCategory(category) : void
+getSendDate() : Date
+getPostID() : string

ExchangePost
-offeredClass : string
-offeredSection : int
-requestedClass : string
-requestedSection : int
+getRequestedClass() : string
+setRequestedClass(requestedClass) : void
+getOfferedClass() : string
+setOfferedClass(offeredClass) : string
+getOfferedSection() : int
+setOfferedSection(offeredSection) : void
+getRequestedSection() : int
+setRequestedSection(requestedSection) : void

BorrowingPost
-requestedItemName : string
-borrowPeriod : Duration
+getRequestedItemName() : string
+setRequestedItemName(requestedItemName) : void
+getBorrowPeriod() : Duration
+setBorrowPeriod(borrowPeriod) : void

ForumPost
-entries : Entry[]
-image : Image
+getEntries() : Entry[]
+setEntries(entries) : void
+getImage() : Image
+setImage(image) : void

DonationPost
-photo : Image
-donatedItemName : string
+getPhoto() : Image
+setPhoto(photo) : void
+getDonatedItemName() : string
+setDonatedItemName(donatedItemName) : void

SecondhandPost
-price : double
-photo : Image
+getPrice() : double
+setPrice(price) : void
+getPhoto() : Image
+setPhoto(photo) : void

LostFoundPost
-where : string
-when : Date
-photo : Image
-status : LostOrFound
-isLost : boolean
+getWhere() : string
+setWhere(where) : void
+getWhen() : Date
+setWhen(when) : void
+getPhoto() : Image
+setPhoto(photo) : void
+getStatus() : LostOrFound
+setStatus(status) : void
+getIsLost() : boolean
+setIsLost(isLost) : void

Entry
-netVote : int
-content : Content
-user : User
-associatedPost : ForumPost
-sendDate : Date
-entryID : string
+downVote() : void
+upVote() : void
+getNetVote() : int
+getContent() : Content
+setContent(content) : void
+getUser() : User
+getAssociatedPost() : ForumPost
+setAssociatedPost(associatedPost) : void

User Interface Management Layer

RegisterPage
+register(user)
+direct ToLoginPage()

LoginPage
+loginRequirement(id, password)
+direct ToForgotPassword()
+direct ToRegisterPage()
+displayBadCredentials()

ForgotPasswordPage
+sendEmail(email)

NotificationPage
+getNotifications() : Notification
+removeNotification(notificationId)
+removeNotifications()

ReportViewPage
+resolveReport(reportID)
+getReports() : Report[]

SectionExchangePage
+addSectionExchangePost(configuration)
+editSectionExchangePage(configuration)
+removeSectionExchangePost(id)

DonatePage
+addDonationPost(configuration)
+editDonationPost(configuration)
+removeDonationPost(id)

NavigationBar
+direct ToProfilePage()
+direct ToDonatePage()
+direct ToForumPage()
+direct ToBorrowPage()
+direct ToLostAndFoundPage()
+direct ToSecondHandPage()
+direct ToSectionExchangePage()
+logout()
+getNotifications() : Notification[]
+direct ToReportPage()

PostPage
+getPostList() : Post[]
+getCurrentPost() : Post
+isUserAdmin() : boolean
+isUserOwner() : boolean
+removeOwnPost()
+banUser()
+forceRemovePost()
+searchpost(Title) : Post[]
+sortByDate(reverse)
+filterByDate(configuration)
+filterByCategory(category)
+direct ToMessagePage()
+direct ToSendReportPage()

ForumPage
+addForumPost(configuration)
+editForumPost(configuration)
+removeForumPost(id)
+addEntry(text)

MessagingPage
+sendMessage(text)
+viewMessages(user)

SendReportPage
+reportPost(postID, description)

LostAndFoundPage
+addLostAndFoundPost(configuration)
+editLostAndFoundPost(configuration)
+removeLostAndFoundPost(id)

SecondhandPage
+addSecondhandPost(configuration)
+editSecondhandPost(configuration)
+removeSecondhandPost(id)
+filterByValue(configuration)
+sortByValue(reverse)

BorrowPage
+addBorrowPost(configuration)
+editBorrowPost(configuration)
+removeBorrowPost(id)

EditProfilePage
+uploadProfilePicture(image)
+saveChanges()
+changePassword(newPassword)
+changeDescription(newDescription)

ProfilePage
+getProfile() : Profile
+direct ToMessagePage()
+banUser()
+direct ToEditProfilePage()
+direct ToSendReportPage()

directed   directs   shows   messages   edits
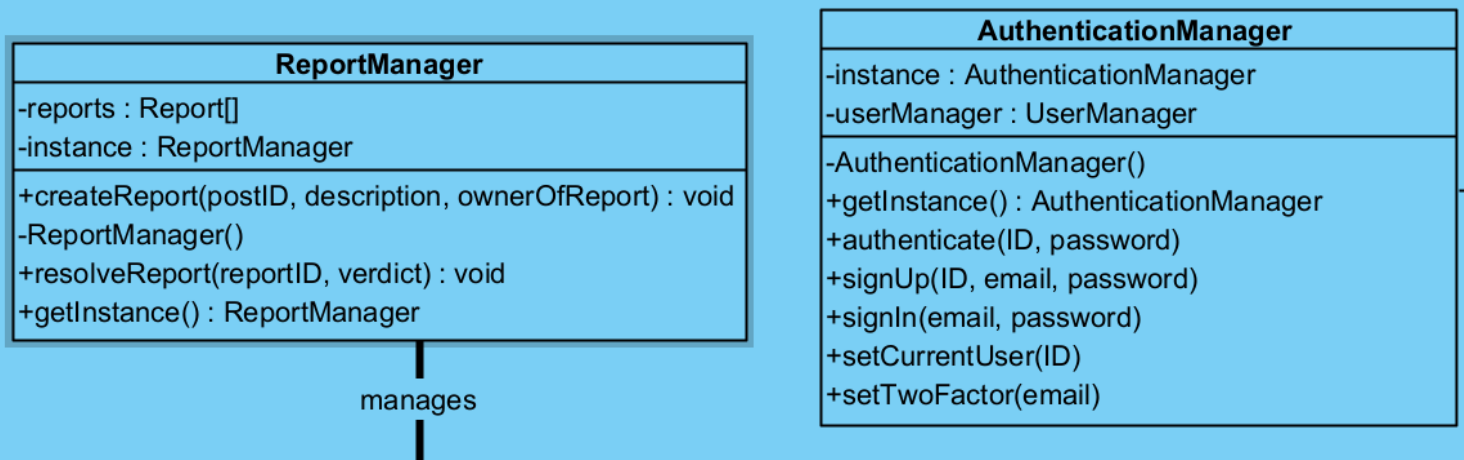
Design Patterns

Facade Pattern

In essence, the Facade pattern acts as a "facade" or a front-facing interface that shields the user from the intricate workings of the system behind it. It allows users to interact with the system through a simpler interface without needing to understand its internal intricacies, thus promoting easier usage and maintenance of the codebase.

We apply the Facade design in two distinct places. One is located in the application layer, and the other is in the database interface layer. The database interface layer consists of several classes known as services, such as SecondHandPostServices, ForumPostServices, etc. Services decouple each endpoint's implementation of the requests. In this way, in the event that the implementation changes, controllers such as SecondHandPostController, ForumPostController, etc., will still function with, at best, minor adjustments. The communication with the database is managed via services' classes. As a result, controller classes don't use any database-related logic.

Singleton Pattern

ReportManager
-reports : Report[]
-instance : ReportManager
+createReport(postID, description, ownerOfReport) : void
-ReportManager()
+resolveReport(reportID, verdict) : void
+getInstance() : ReportManager

manages

AuthenticationManager
-instance : AuthenticationManager
-userManager : UserManager
-AuthenticationManager()
+getInstance() : AuthenticationManager
+authenticate(ID, password)
+signUp(ID, email, password)
+signIn(email, password)
+setCurrentUser(ID)
+setTwoFactor(email)

Singleton pattern plays a crucial role in ensuring that a class has only one instance and provides a global point of access to it. The Singleton pattern is particularly beneficial when there is a need for a single, shared instance of a class that controls actions, coordinates activities or manages resources. By enforcing a single instance, the Singleton pattern helps maintain consistency and prevents unnecessary duplication of resources. In our implementation of the Singleton pattern, we have encapsulated the instantiation process within the class itself through a private constructor, ensuring that clients cannot create additional instances of the respective class. The two classes that utilize the creational design pattern of Singleton are ReportManager and AuthenticationManager. In ReportManager, we use it in reporting the posts to prevent all users from accessing the report content. AuthenticationManager, we will use it to evaluate the user activities. Only a single instance of them is necessary in our system. Thus, we implemented them in such a manner.