CS 319 TERM PROJECT

Section 3
Group 3B
UNICLAPP

Design Report

Project Group Members:

1. Ayberk Yaşa	21801847
2. Fatih Kaplama	21802755
3. Görkem Ayten	21802399
4. Mert Atakan Onrat	21802520

Supervisor: Eray Tüzün

1. Introduction

1.1 Purpose of the System

This project is a web-based student club application. This application basically allows the clubs and students at Bilkent University to communicate more interactively. In this application, clubs can open their own accounts and make event announcements from here. Students, on the other hand, can follow these clubs using their own accounts and be informed about these announcements easily. In addition, the application offers students the opportunity to evaluate the clubs and the events organized by the clubs. In this way, each event and club will have a point and the clubs will be ranked.

1.2 Design Goals

1.2.1 Performance

A common problem with websites is that they run slowly when instant traffic increases. It is undesirable for this project to run slowly in such a situation. Therefore, all factors that can slow down the response time of the system are avoided. The project is deployed on the fastest and most scalable server. External libraries, which may have a negative impact on the working speed of the project, are used as little as possible. In other words, up to 15000 users, which is the predictable number of simultaneous users, it is designed to keep the loading time of the pages under 1 second and the response time of the buttons around 0.1 seconds.

1.2.2 Usability

Since student clubs usually announce their events via email, it is difficult for students to follow these events. In addition, students have to check emails one by one to find information about an event. This situation makes their job significantly more difficult and is undesirable. The application interface, buttons and texts are chosen quite simply for students to do these operations more easily. Students can easily find out which club has which event in just a few steps. In this way, even students who are introduced to the application for the first time can easily adapt to this environment without difficulty. In addition, there is a tutorial on the application interface that explains what features the application has and how these features can be used.

1.2.3 Reliability

The program is designed and expected to work and behave in a way that the number of times the errors or bugs occur in low percentage while users are using the application. The unwanted system problems are minimized to ensure that the users can use the application fluently. Users must access their upcoming and past events information %99 of the time without failure and users who close the browser without properly logging out of their account will be automatically logged out. By this way the reliability of the program is increased and users are expected not to stop using the application immediately after encountering constant errors.

1.2.4 Scalability

Since this project is a student and student club interaction application, many students and club officials may log into the website during the day. That's why scalability is very important for this project. The server will be reserved in such a way that the bandwidth limit is not exceeded until the daily total, not simultaneous, of 50000 visitors.

2. High-level Software Architecture

2.1 Subsystem Decomposition

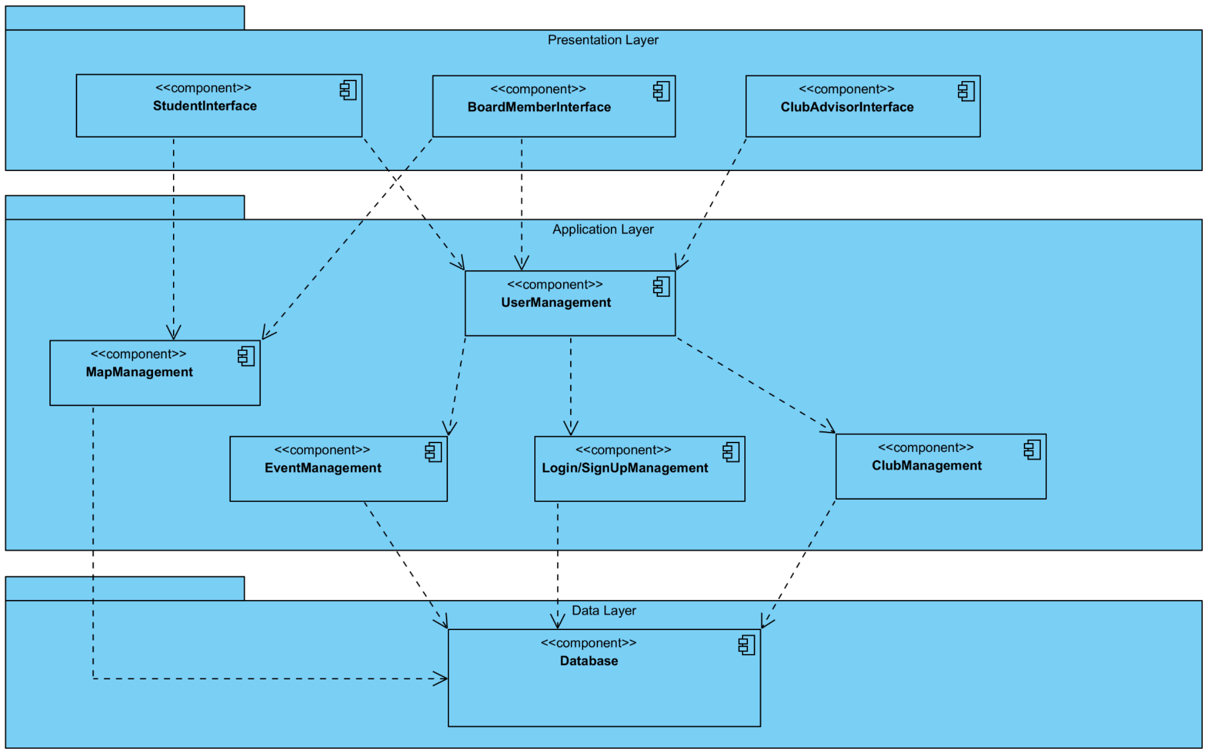

Presentation Layer
<<component>> StudentInterface
<<component>> BoardMemberInterface
<<component>> ClubAdvisorInterface
Application Layer
<<component>> UserManagement
<<component>> MapManagement
<<component>> EventManagement
<<component>> Login/SignUpManagement
<<component>> ClubManagement
Data Layer
<<component>> Database

Figure 1: *Subsystem Decomposition of the System*

In this project, the three layer architecture consisting of presentation, application, and data layer is decided to be used. The Data Layer consists of the Database subsystem containing objects that need to be stored in the project. This subsystem is called by four of the subsystems in the Application Layer, which are MapManagement, EventManagement, Login/SignUpManagement, and ClubManagement.

The Application Layer contains five different subsystems which are UserManagement, MapManagement, EventManagement, Login/SignUpManagement, and ClubManagement. All these subsystems in this layer are responsible for the operations that process the data obtained from the Database subsystem or each of the subsystems in the Presentation Layer. Moreover, each subsystem is constrained to certain functionalities. Thus, thanks to all subsystems, a structure containing all the functionality on the back-end is created. Lastly, the subsystems in this layer call the Database subsystem.

The Presentation Layer consists of three different weakly coupled subsystems containing the components of the user interface. The reason why it is divided into three different subsystems is that there are three different user types in the project and each of these user types accesses a different interface. Therefore, each subsystem contains all the pages that the user who is specific to that subsystem can access, and the functionality that those pages offer to the user. Lastly, the subsystems in this layer call the subsystems in the Application Layer.

2.1.1 Presentation Layer

2.1.1.1 StudentInterface

This subsystem creates the student interface for the application. The student interface consists of the main page, profile page, explore page, followings page, upcoming events page, event history page, event tracker page, campus map page, and leaderboard page. Template design pattern is used during the implementation of student interface.

2.1.1.2 BoardMemberInterface

This subsystem creates the board member interface for the application. The board member interface consists of the main page, profile page, explore page, followings page, upcoming events page, event history page, event tracker page, campus map page, leaderboard page, club profile page, organiz event page, and event result page. If the board member is also board chairman, the board chairman can see an additional page named rank board member. Template design pattern is used during the implementation of board member interface.

2.1.1.3 ClubAdvisorInterface

This subsystem creates the club advisor interface for the application. The club advisor interface consists of the main page, profile page different from others, club profile page, and event result page. Since the club advisor has fewer rights than others in this application, s/he access fewer pages than others. Template design pattern is used during the implementation of club advisor interface.

2.1.2 Application Layer

2.1.2.1 UserManagement

UserManagement subsystem is responsible for the business logic of the user. This subsystem has BoardMemberController, StudentController, ClubAdvisorController, BoardChairmanController classes. These classes perform operations such as updating, deleting, adding, removing the user object.

2.1.2.2 ClubManagement

ClubManagement subsystem is responsible for the business logic of the club. This subsystem has a ClubController class. This class performs operations such as updating, deleting, adding, and removing the club object. Thanks to this class, club objects are manipulated in accordance with the request coming from the user.

2.1.2.3 EventManagement

EventManagement subsystem is responsible for the business logic of the event. This subsystem has EventController, EvaluationController. These classes perform operations such as updating, deleting, adding, and removing the event object. Thanks to these classes, event objects are controlled in accordance with the request coming from the user.

2.1.2.4 Login/SignUpManagement

Login/SignUpManagement subsystem is responsible for the business logic of the act of login and sign up. This subsystem has a UserLoginSignUpManagement class. This class controls users' login and logout to the system. When a user tries to register to the system, it makes the necessary checks and, if appropriate, registers the user to the system. It performs the authentication process during login.

2.1.2.5 MapManagement

MapManagement subsystem is responsible for the business logic of the campus map. This subsystem has a CampusMapController class. This class performs operations such as updating, deleting, adding, and removing the campus map presented to the user. It is created once so that the campus map object is not re-rendered each time the user wants to see the campus map. That's why the singleton design pattern is used.

2.1.3 Data Layer

2.1.3.1 Database

Database subsystem is responsible for storing the entity objects in the application. These entity objects to be stored in this subsystem are User, Student, BoardMember, BoardChairman, ClubAdvisor, OEM, Club, Event, Evaluation, Classroom, Place, Building, Image, Campus Map and LeaderBoard. Thanks to this subsystem, the result of the changes made by the subsystems in the application layer will be stored simultaneously. Thanks to the bidirectional communication with the application layer, both write and read operations will be performed.

2.2 Deployment Diagram

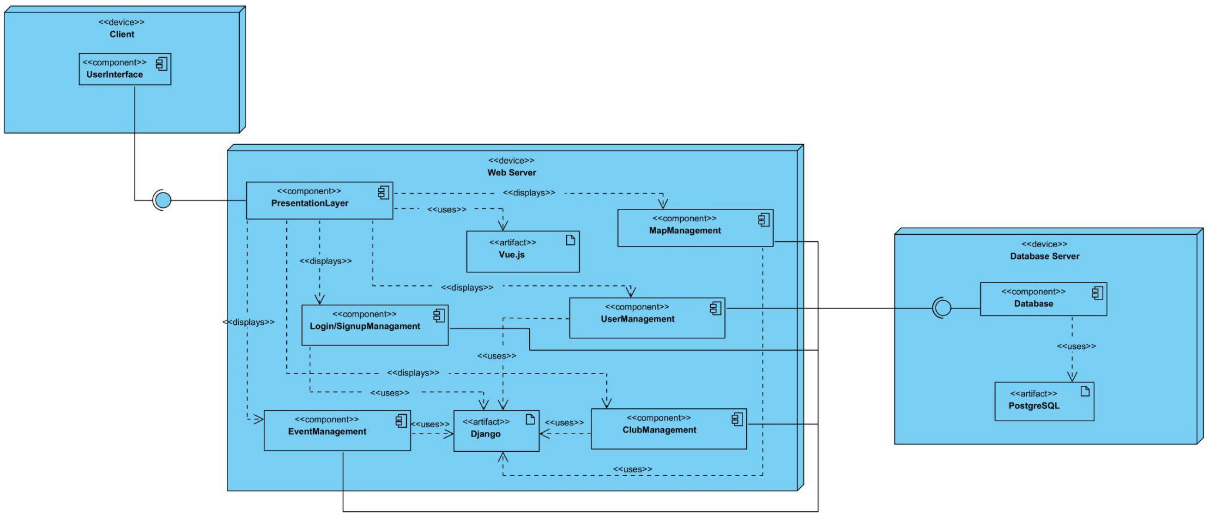


Figure 2: *Deployment Diagram of the System*

This diagram demonstrates the physical hardware which the software system will execute on. This project has three nodes named Client, Web Server, and Database Server. Client node which is a PC includes only a UserInterface component. The UserInterface component in the Client node is connected to the PresentationLayer component in the Web Server node with provided and required interfaces. Web Server node has six components and two artifacts. These components are PresentationLayer, UserManagement, MapManagement, EventManagement, ClubManagement, and Login/SignupManagement. These artifacts are Vue.js which is the progressive framework of JavaScript and Django which is the web framework of Python. The PresentationLayer component is connected to other components with dependency whose type is display. All components except for PresentationLayer in Web Server node uses the artifact named Django, while PresentationLayer uses the artifact named Vue.js. All components except for PresentationLayer in the Web Server node are connected to the Database component in the Database Server node with provided and required interfaces. Database Server node has one component and one artifact. This component is Database and this artifact is PostgreSQL which is an open source object-relational database. The Database component uses the artifact named PostgreSQL so that it stores data coming from components located on the Web Server node.

2.3 Hardware/Software Mapping

This student club manager application does a hardware component which is a production server to run successfully on the web browsers. This production server has 2 vCPUs, 2 GB RAM, and 16 GB storage memory and is in Amazon EC2 cloud server. This production server is run on the Linux operating system. Moreover, a web browser is required on users' hardware systems such as computers and phones. This web browser can be Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, or Safari [1-5]. For Google Chrome, the browser version should be 96.0.4664.45 or later. For Microsoft Edge, the browser version should be 96.0.1054.34 or later. For Mozilla Firefox, the browser version should be 94.0.2 or later. For Opera, the browser version should be 81.0.4196.54 or later. For Safari, the browser version should be 15.0 or later. Moreover, the version of Django to be used for the back-end is 3.2 and Vue.JS to be used for the front-end has the version of 2.x [6-7].

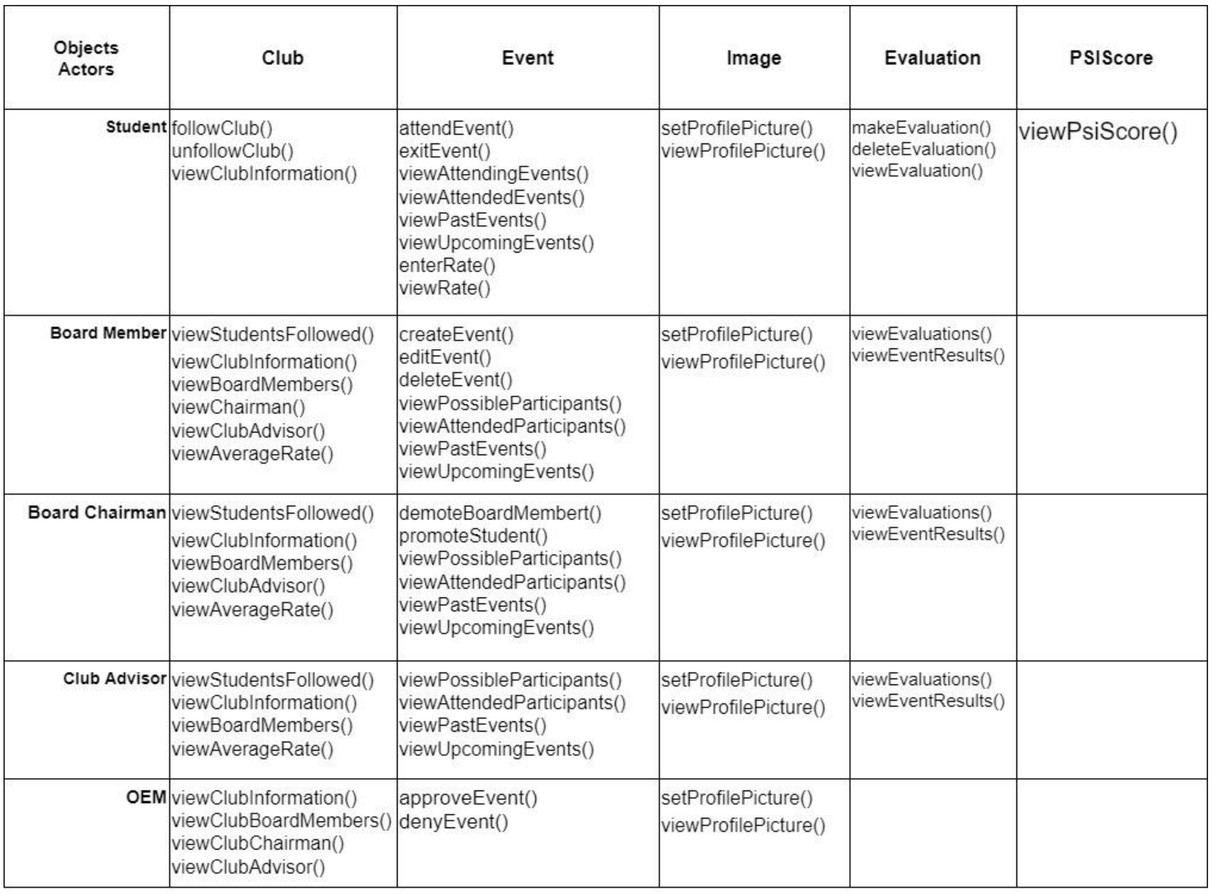
2.4 Persistent Data Management

In this project, a specific database is needed since clients send a request to the server to access the specific information such as student clubs, events or profile. These data should be accessed quickly. In this project, PostgreSQL [8] will be used. The reason PostgreSQL is chosen, it is an object relational database that is convenient to apply OOP principles. Moreover, PostgreSQL has various functionalities such as storing arrays. Thanks to PostgreSQL, the project's database will be more functional, speed and safe. Project's main objects such as User, Student, BoardMember, BoardChairman, ClubAdvisor, OEM, Club, Event, Evaluation, Classroom, Place, Building, Image, Campus Map and LeaderBoard will be stored in this database that is deployed in a cloud server by using AWS [9]. Clients can make some operations such as sending GET, PUT, POST and DELETE requests to the server. It is important to provide such operations to client.

2.5 Access Control and Security

Our web application has access control and it provides security. It is provided access control by giving permission to the logged user with the matching used type. In this way, a user cannot use the properties of a user type that they do not belong to. Also, to improve the security of the web application, there will be a good password policy. Users must create a password of at least 8 characters including lowercase, uppercase, special characters and numbers. These improve the security and the access control of the application.

Matrix for access control



Objects Actors	Club	Event	Image	Evaluation	PSIScore

Student	followClub()
	unfollowClub()
	viewClubInformation()
		attendEvent()
		exitEvent()
		viewAttendingEvents()
		viewAttendedEvents()
		viewPastEvents()
		viewUpcomingEvents()
		enterRate()
		viewRate()
			setProfilePicture()
			viewProfilePicture()
				makeEvaluation()
				deleteEvaluation()
				viewEvaluation()
					viewPsiScore()

Board Member	viewStudentsFollowed()
	viewClubInformation()
	viewBoardMembers()
	viewChairman()
	viewClubAdvisor()
	viewAverageRate()
		createEvent()
		editEvent()
		deleteEvent()
		viewPossibleParticipants()
		viewAttendedParticipants()
		viewPastEvents()
		viewUpcomingEvents()
			setProfilePicture()
			viewProfilePicture()
				viewEvaluations()
				viewEventResults()	

Board Chairman	viewStudentsFollowed()
	viewClubInformation()
	viewBoardMembers()
	viewClubAdvisor()
	viewAverageRate()
		demoteBoardMembert()
		promoteStudent()
		viewPossibleParticipants()
		viewAttendedParticipants()
		viewPastEvents()
		viewUpcomingEvents()
			setProfilePicture()
			viewProfilePicture()
				viewEvaluations()
				viewEventResults()	

Club Advisor	viewStudentsFollowed()
	viewClubInformation()
	viewBoardMembers()
	viewAverageRate()
		viewPossibleParticipants()
		viewAttendedParticipants()
		viewPastEvents()
		viewUpcomingEvents()
			setProfilePicture()
			viewProfilePicture()
				viewEvaluations()
				viewEventResults()	

OEM	viewClubInformation()
	viewClubBoardMembers()
	viewClubChairman()
	viewClubAdvisor()
		approveEvent()
		denyEvent()
			setProfilePicture()
			viewProfilePicture()		

2.6 Boundary Conditions

2.6.1 Initialization

Since this is a web-based application, no installation is required. All that is required for the application to work is to be connected to the internet. Therefore, the operation of the application is quite simple. Since this project is an application whose frontend and backend are linked, the connection between the frontend and the backend must be established for the project to work properly on the development server. After this connection is established, admin uses pm2, which is the process manager that helps to manage the application online, to initialize the application on the development server.

2.6.2 Termination

This project is an application whose frontend and backend are linked. Therefore, if the admin terminates the frontend, it means the backend is also terminated or vice versa. Therefore, when the admin initiates a termination event, all the data of the users and events such as registering for an event, creating an event, following a club, etc., performed by the user within the application are preserved in the database the database will be in constant interaction with the application. Admin uses pm2, which is the process manager that helps to manage the application online, to terminate the application on the development server.

2.6.1 Failure

Amazon web service (AWS) will be used so that this application can work remotely and the database can be used and updated continuously. If there is a server-related problem in the application, this issue will be forwarded to the authorized persons. In case of bugs or errors that may occur in the application, the data will never be lost because it is saved in the database.

3. Low-level Design

3.1 Object Design Trade-offs

Maintainability versus Performance: This web application has many layers for almost every part of the project and this makes some disadvantages on performance. However, comparing advantages of maintainability with disadvantages of the performance of the application, the benefits of maintainability are much more.

Security versus Usability: This web application makes changing password in each three months obligatory. Each user has to create a new password different from the last three passwords in three months. Although this feature makes this application more secure, it decreases the usability.

Functionality versus Usability: This web application is designed especially for the students and student clubs of Bilkent University. This web application offers different functionality for each user such as rating for students and leaderboard for student clubs. Therefore, while the number of users grows, the usability of the web application would become complicated for new users.

3.2 Final Object Design

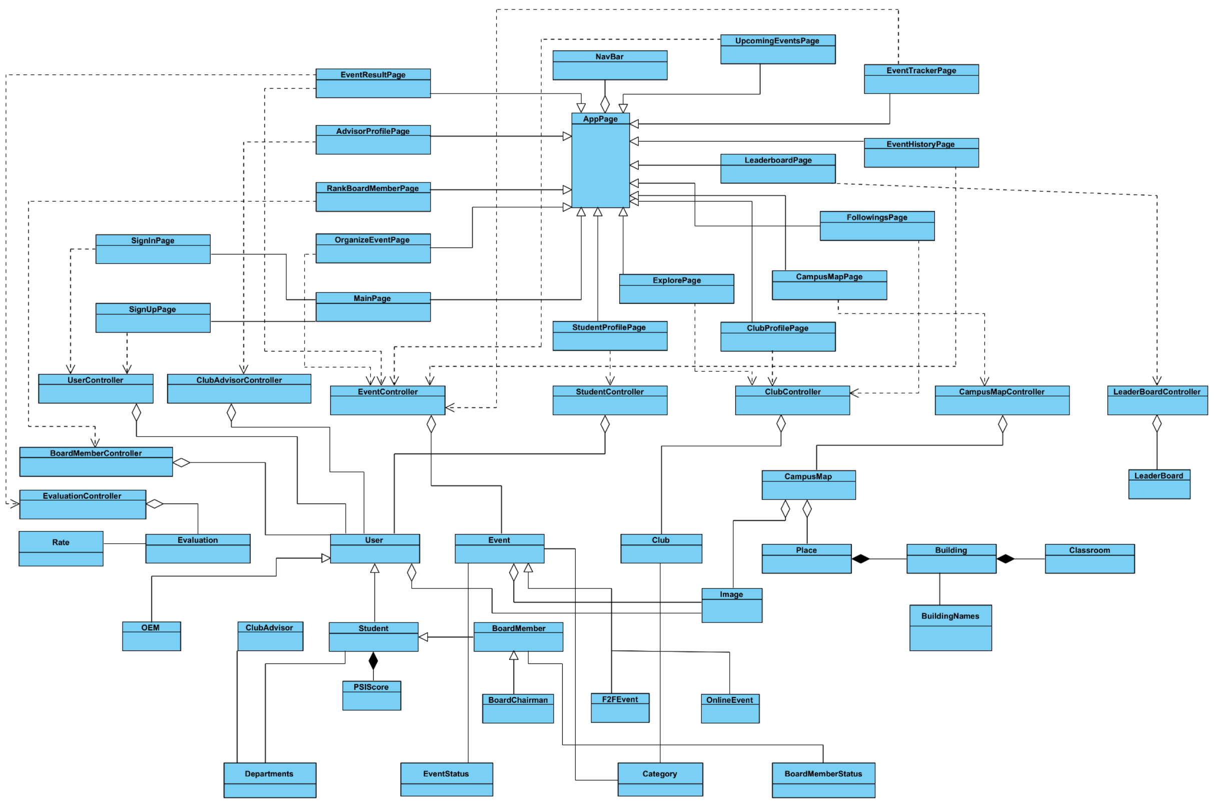


Figure 3: *Abstraction of Final Object Design*

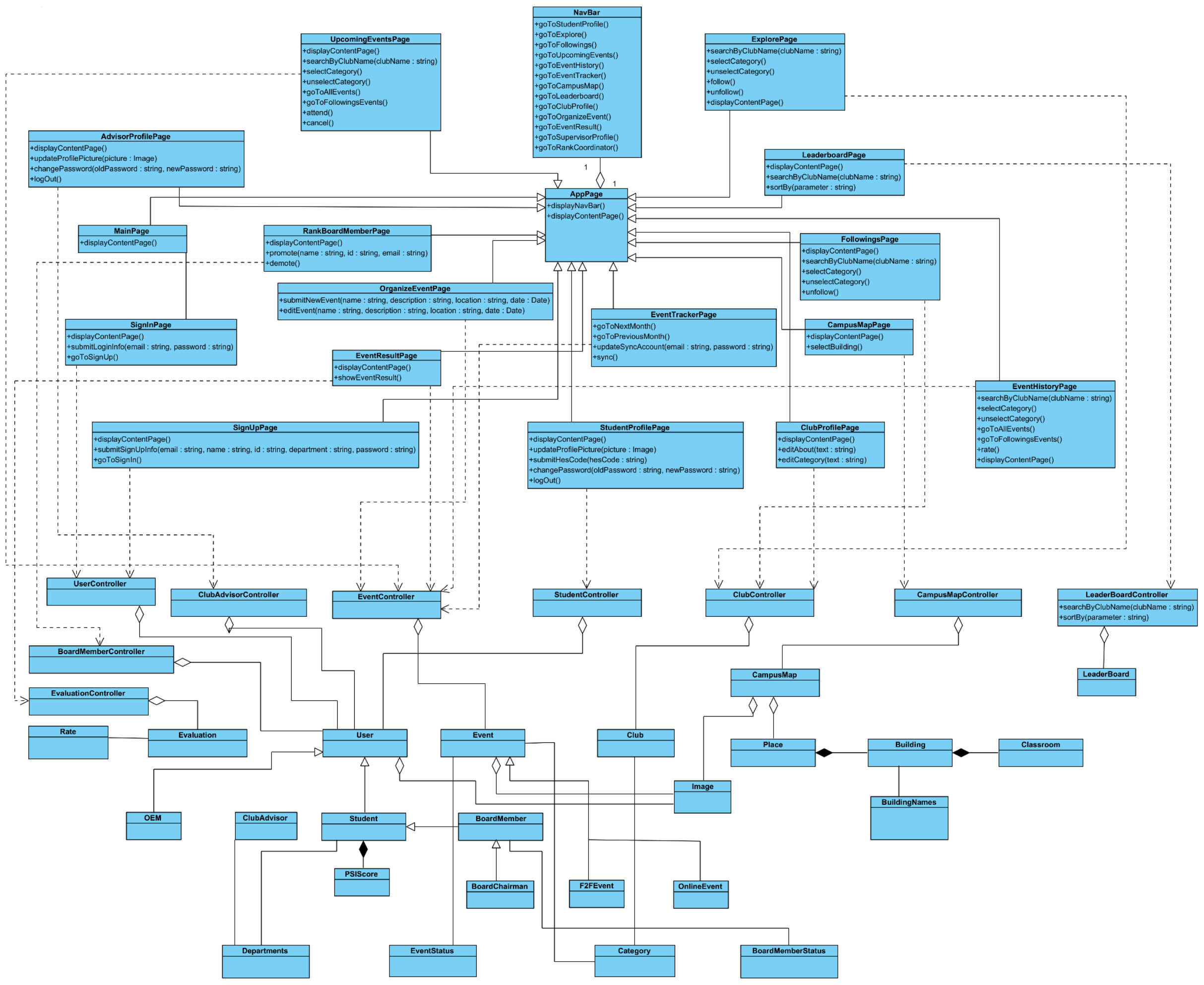


Figure 4: *Classes of Boundary Objects*

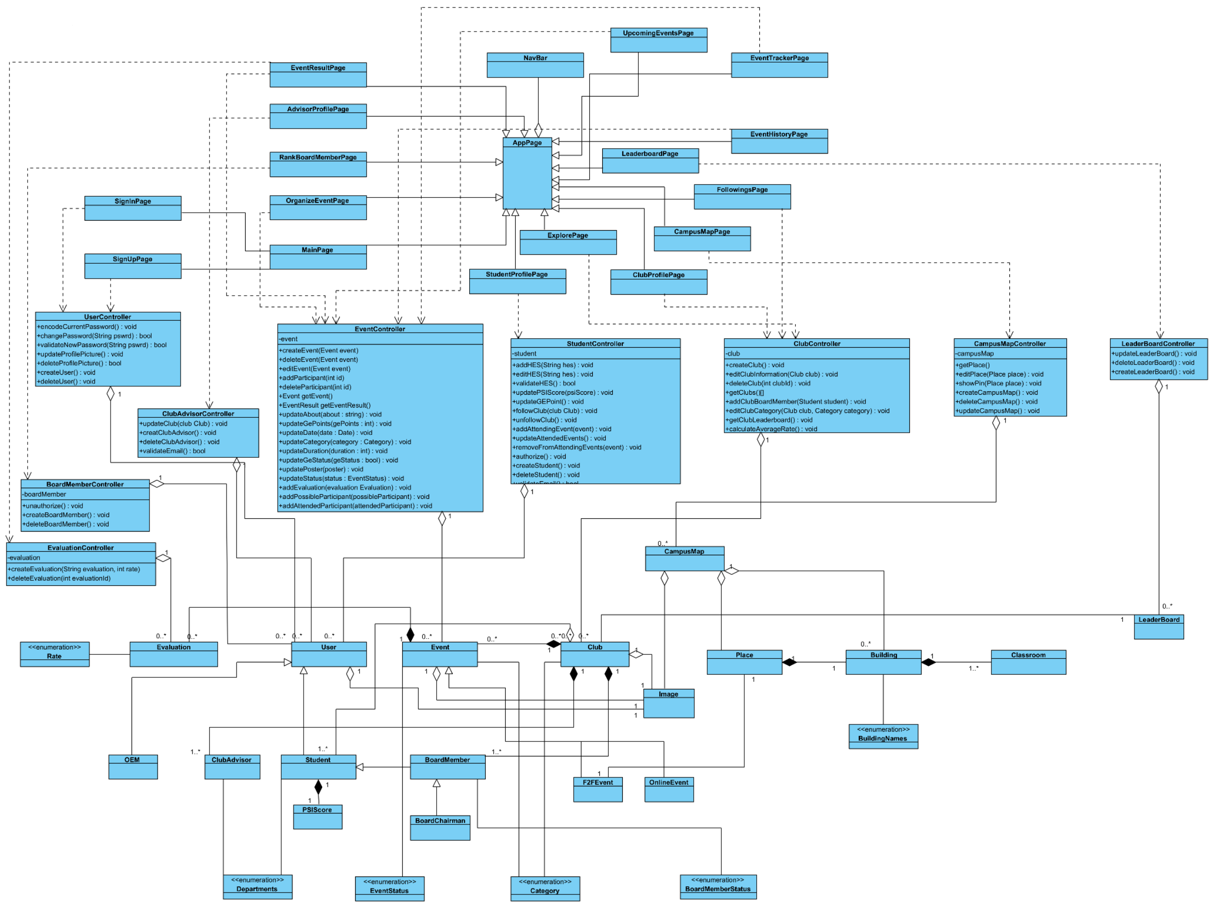

UpcomingEventsPage
EventTrackerPage
NavBar
EventResultPage
AdvisorProfilePage
EventHistoryPage
AppPage
LeaderboardPage
RankBoardMemberPage
FollowingsPage
OrganizeEventPage
SignInPage
ExplorePage
CampusMapPage
MainPage
SignUpPage
StudentProfilePage
ClubProfilePage
UserController
+encodeCurrentPassword() : void
+changePassword(String pswrd) : bool
+validateNewPassword(String pswrd) : bool
+updateProfilePicture() : void
+deleteProfilePicture() : bool
+createUser() : void
+deleteUser() : void
EventController
-event
+createEvent(Event event)
+deleteEvent(Event event)
+editEvent(Event event)
+addParticipant(int id)
+deleteParticipant(int id)
+Event getEvent()
+EventResult getEventResult()
+updateAbout(about : string) : void
+updateGePoints(gePoints : int) : void
+updateDate(date : Date) : void
+updateCategory(category : Category) : void
+updateDuration(duration : int) : void
+updateGeStatus(geStatus : bool) : void
+updatePoster(poster) : void
+updateStatus(status : EventStatus) : void
+addEvaluation(evaluation Evaluation) : void
+addPossibleParticipant(possibleParticipant) : void
+addAttendedParticipant(attendedParticipant) : void
StudentController
-student
+addHES(String hes) : void
+editHES(String hes) : void
+validateHES() : bool
+updatePSIScore(psiScore) : void
+updateGEPoint() : void
+followClub(club Club) : void
+unfollowClub() : void
+addAttendingEvent(event) : void
+updateAttendedEvents() : void
+removeFromAttendingEvents(event) : void
+authorize() : void
+createStudent() : void
+deleteStudent() : void
ClubController
-club
+createClub() : void
+editClubInformation(Club club) : void
+deleteClub(int clubId) : void
+getClubs()[]
+addClubBoardMember(Student student) : void
+editClubCategory(Club club, Category category) : void
+getClubLeaderboard() : void
+calculateAverageRate() : void
CampusMapController
-campusMap
+getPlace()
+editPlace(Place place) : void
+showPin(Place place) : void
+createCampusMap() : void
+deleteCampusMap() : void
+updateCampusMap() : void
LeaderBoardController
+updateLeaderBoard() : void
+deleteLeaderBoard() : void
+createLeaderBoard() : void
ClubAdvisorController
+updateClub(club Club) : void
+creatClubAdvisor() : void
+deleteClubAdvisor() : void
+validateEmail() : bool
BoardMemberController
-boardMember
+unauthorize() : void
+createBoardMember() : void
+deleteBoardMember() : void
EvaluationController
-evaluation
+createEvaluation(String evaluation, int rate)
+deleteEvaluation(int evaluationId)
CampusMap
LeaderBoard
<<enumeration>>
Rate
Evaluation
User
Event
Club
Place
Building
Classroom
Image
<<enumeration>>
BuildingNames
OEM
ClubAdvisor
Student
BoardMember
F2FEvent
OnlineEvent
PSIScore
BoardChairman
<<enumeration>>
Departments
<<enumeration>>
EventStatus
<<enumeration>>
Category
<<enumeration>>
BoardMemberStatus

Figure 5: *Classes of Controller Objects*

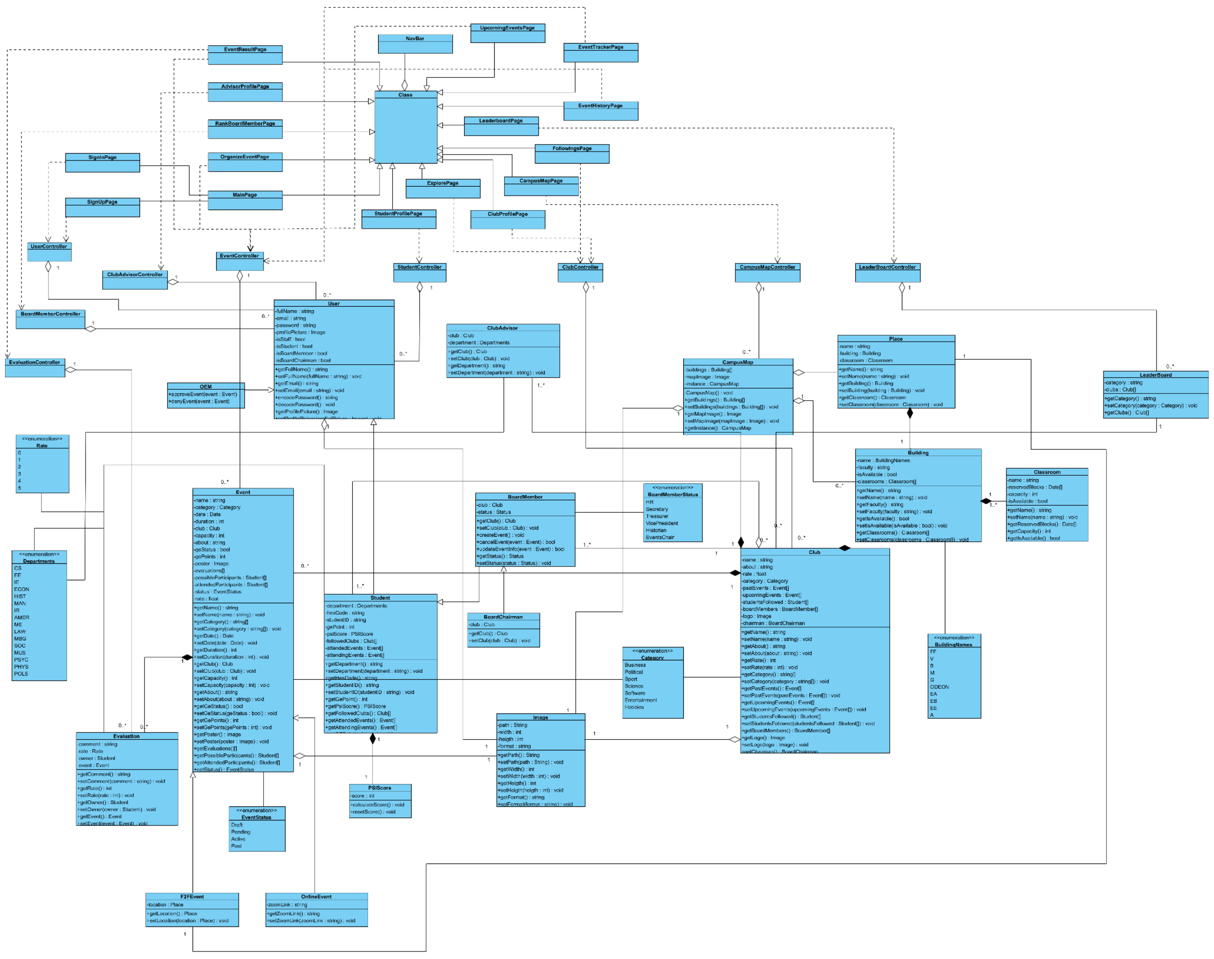


Figure 6: *Classes of Entity Objects*

3.3 Design Patterns

3.3.1 Template Design Pattern

In our web application some methods of components must be run on all pages. Therefore we have decided to use Template Design Pattern and we have written these methods in a template. After that, since we have a template including all common codes that some pages must use, we have used this template in these pages. Therefore, we have avoided the code duplication and we have used the same template by writing only once. In conclusion, the template design pattern has increased the maintainability of the project and avoided code clutter.

3.3.2 Singleton Design Pattern

In our project some classes must have one instance and this instance should be used everywhere. Therefore, we have decided to use the Singleton Design Pattern. We have changed the type of the constructor of the class that must have one instance as a private and we have added the static instance variable and getInstance method. Therefore, if the instance of this class has never been created, it has created the new instance, if the instance of this class has been created before, it has returned the instance created before. As a result, since the instance of this class has been created only once, the singleton design pattern has prevented the creation of extra objects and tiring the system and database.

3.4 Packages

3.4.1 src.boundary Package

Boundary has no classes since it only creates a namespace.

3.4.2 src.boundary.studentInterface Package

This package contains the classes which are related to the user interface from the student's perspective.

3.4.3 src.boundary.boardMemberInterface Package

This package contains the classes which are related to the user interface from the board member's perspective.

3.4.4 src.boundary.advisorInterface Package

This package contains the classes which are related to the user interface from the club advisor's perspective.

3.4.5 src.controller Package

Controller has no classes since it only creates a namespace.

3.4.6 src.controller.userManagement Package

This package includes the controller classes for the operations of the management of the user information.

3.4.7 src.controller.eventManagement Package

This package includes the controller classes for the operations of the management of the event information.

3.4.8 src.controller.clubManagement Package

This package includes the controller classes for the operations of the management of the student club information.

3.4.9 src.controller.mapManagement Package

This package includes the controller classes for the operations of the management of the campus map information.

3.4.10 src.controller.signInSignUpManagement Package

This package includes the controller classes for the operations of the management of the authentication information.

3.4.11 src.entity Package

Entity has no classes since it only creates a namespace.

3.4.12 src.entity.user Package

This package contains classes of three user types which are Student, BoardMember, and ClubAdvisor and the related classes to the user classes.

3.4.13 src.entity.event Package

This package contains the creation and management of event objects and other related classes.

3.4.14 src.entity.club Package

This package contains the student club class and the helper classes for this one.

3.4.15 src.entity.map Package

This package contains the classes related to the campus map such as place, building, classroom and the management of all these objects.

3.5 Class Interfaces

3.5.1 Boundary Classes

3.5.1.1 NavBar Class

Explanation: This class includes the main operations buttons of the application.

Methods:
goToStudentProfile(): This method leads the user to enter his/her profile page.

goToExplore(): This method allows users to travel to the exploring clubs page. There are clubs on that page.

goToFollowings(): This method allows the user to go to the page including clubs followed by that user.

goToUpcomingEvents(): This method allows the user to visit the upcoming events page. In that page this user can see the all events or events performed by the club this user is following.

goToEventHistory(): This method opens the event history page on the screen and allows the user to visit this page.

goToEventTracker(): This method allows the user to enter the event tracker page and displays this page on the screen.

goToCampusMap(): This method helps the user enter the campus map page and opens this page on the screen.

goToLeaderboard(): This method opens the leaderboard page and helps the user examine the leaderboard of clubs.

goToClubProfile(): This method displays the club profile page for club advisor and board member.

goToOrganizeEvent(): This method allows the board member to organize an event. Firstly this method opens the organize event page.

goToEventResult(): This method leads the users to see the specific event result and displays the event result page on the screen.

goToAdvisorProfile(): This method helps the club advisor enter his/her profile page.

goToRankBoardMember(): This method allows the club advisor to change the status of a student. The club advisor can promote a student to board member or can demote a board member to a student.

3.5.1.2 SignInPage Class

Explanation: This class represents the user interface of a page that includes the student identification number and password areas for users to login.

Methods:
submitLoginInfo(email: string, password: string): This method checks the email and password data entered whether there is such a user on the database or not.
goToSignUp(): This method changes the screen from sign in to sign up..

3.5.1.3 SignUpPage Class

Explanation: This class represents the user interface of a page that includes the name, identification number, department password and email areas for the users to create an account.

Methods:
submitSignUpInfo(email: string, name: string, id: string, department: string, password: string): This method submits the information entered by the user to the database and creates a new account on the database.
goToSignIn(): This method changes the screen from sign up to sign in.

3.5.1.4 StudentProfilePage Class

Explanation: This class represents the user interface of a page that includes the profile photograph, name, id, email, department, PSI score, and HES code of the user.

Methods:
updateProfilePicture(picture: Image): This method allows the user to change his/her profile photograph.
submitHesCode(hesCode: string): This method allows the user to submit the HES code to the database.
changePassword(oldPassword: string, newPassword: string): This method allows the user to change his/her password.
logOut(): This method allows the user to log out and end the session.

3.5.1.5 ExplorePage Class

Explanation: This class represents the user interface of a page that includes a list of the clubs registered on the system.

Methods:
searchByClubName(clubName: string): This method helps the user search the clubs by entering the club’s name.
selectCategory(): This method allows the user to select a category of clubs for filter operation.
unselectCategory(): This method is used to cancel the selection when the selected category is clicked again.
follow(): This method allows the user to follow clubs and adds that club to the list of clubs the user follows.
unfollow(): This method allows the user to unfollow the clubs and removes that club from the list of clubs the user follows.

3.5.1.6 FollowingsPage Class

Explanation: This class represents the user interface of a page that includes the list of clubs followed by the student entered this page.

Methods:
searchByClubName(clubName: string): This method helps the user search the clubs by entering the club's name.
selectCategory(): This method allows the user to select a category of clubs for filter operation.
unselectCategory(): This method is used to cancel the selection when the selected category is clicked again.
unfollow(): This method allows the user to unfollow the clubs and removes that club from the list of clubs the user follows.

3.5.1.7 UpcomingEventsPage Class

Explanation: This class represents the user interface of a page that includes the events of the clubs.

Methods:
searchByClubName(clubName: string): This method helps the user search the clubs by entering the club's name.
selectCategory(): This method allows the user to select a category of events for filter operation.
unselectCategory(): This method is used to cancel the selection when the selected category is clicked again.
goToAllEvents(): This method allows the user to see all events performed by the clubs registered on the system.
goToFollowingsEvents(): This method allows the user to see events performed by the clubs the user follows.
attend(): This method allows the user to register the event.
cancel(): This method allows the user to cancel registration of the event.

3.5.1.8 EventHistoryPage Class

Explanation: This class represents the user interface of a page that includes the information of events the user attended.

Methods:
searchByClubName(clubName: string): This method helps the user search the clubs by entering the club's name.
selectCategory(): This method allows the user to select a category of events for filter operation.

unselectCategory(): This method is used to cancel the selection when the selected category is clicked again.
goToAllEvents(): This method allows the user to see all events performed by the clubs registered on the system.
goToFollowingsEvents(): This method allows the user to see events performed by the clubs the user follows.
rate(): This method allows the user to rate the events that the user attended out of 5.

3.5.1.9 EventTrackerPage Class

Explanation: This class represents the user interface of a page that includes the calendar showing the date of events.

Methods:
goToNextMonth(): This method moves the month that appears in the calendar on the page to the next month.
goToPreviousMonth(): This method moves the month that appears in the calendar on the page to the previous month.
updateSyncAccount(email: string, password: string): This method allows the user to update the information of Google Account which is synchronized with the event tracker data.
sync(): This method is called by clicking the sync button and starts the synchronization process.

3.5.1.10 CampusMapPage Class

Explanation: This class represents the user interface of a page that includes maps for the user and users can select the building they want to go to and see it on the map.

Methods:
selectBuilding(): This class allows the user to select a building in the campus of Bilkent University on the drop-down menu.

3.5.1.11 LeaderboardPage Class

Explanation: This class represents the user interface of a page that includes the leaderboard of the clubs. Users can filter this leaderboard.

Methods:
searchByClubName(clubName: string): This class allows the user to search the student clubs by name on the leaderboard.
sortBy(parameter: string): This class allows the user to sort the student clubs by a specified parameter on the leaderboard.

3.5.1.12 ClubProfilePage Class

Explanation: This class represents the user interface of a page that includes the profile page of a club. This page can only be accessed by the club advisor and the board member.

Methods:
editAbout(text: string): This method is called by only the board member and allows the board member to edit the general information about the student club.
editCategory(text: string): This method is called by only the board member and allows the board member to edit the category of the club.

3.5.1.13 OrganizeEventPage Class

Explanation: This class represents the user interface of a page that allows the board member to organize an event assigned to the club.

Methods:
submitNewEvent(name: string, description: string, location: string, date: Date): This method enables the board member to submit the information about the new event desired to be created.
editEvent(name: string, description: string, location: string, date: Date): This method enables the board member to edit the already existing event's information.

3.5.1.14 EventResultPage Class

Explanation: This class represents the user interface of a page that contains the past event results.

Methods:
showEventResult(): This method provides the board member and advisor with inspecting the result of the past events.

3.5.1.15 AdvisorProfilePage Class

Explanation: This class represents the user interface of a page that includes the user information of the club advisor.

Methods:
updateProfilePicture(picture: Image): This method provides the advisor with a new profile picture for the advisor's account.
changePassword(oldPassword: string, newPassword: string): This method allows the advisor to change the password.
logOut(): The website can be logged out using this method.

3.5.1.16 RankBoardMemberPage Class

Explanation: This class represents the user interface of a page that includes the promoting and demoting operations by the club advisor.

Methods:
promote(name: string, id: string, email: string): This method allows the club advisor to promote a student to the rank of the board member.
demote(): This method allows the club advisor to demote a board member to the rank of the student.

3.5.1.17 AppPage Class

Explanation: This class represents the general application page and displays the navigation bar and contents..

Methods:
displayNavBar(name: string, id: string, email: string): This method displays the navigation bar and application bar on the screen.
displayContentPage(): This method displays the contents of the page. These contents may be Explore Page, UpcomingEvents Page, Profile Page, etc.

3.5.2 Controller Classes

3.5.2.1 UserController Class

Explanation: This class is responsible for the operations of the User object such as delete, update and create.
Methods:
encodeCurrentPassword(): This method encodes the password.
changePassword(pswrd string): This method is for changing password.
validateNewPassword(pswrd string): This method is for validating the password.
updateProfilePicture(): This method is for updating the profile picture.
deleteProfilePicture(): This method is for deleting the profile picture.
createUser(): This method creates a user.
deleteUser(): This method deletes a user.

3.5.2.2 BoardMemberController Class

Explanation: This class is responsible for the operations of the board member object such as delete, update and create.
Methods:
unauthorize(): This method is for making Board Member a Student.
createBoardMember(): This method is for creating a Board Member.
deleteBoardMember(): This method is deleting a Board Member.

3.5.2.3 ClubAdvisorController Class

Explanation: This class is responsible for the operations of the ClubAdvisor object such as delete, update and create.
Methods:
updateClub(club Club): This method is for updating the club that the Club Advisor belongs to.
createClubAdvisor(): This method is for creating a Club Advisor.
deleteClubAdvisor(): This method is for deleting a Club Advisor.

3.5.2.4 EventController Class

Explanation: This class is responsible for the operations of the Event object such as delete, update and create.
Methods:

3.5.2.5 StudentController Class

Explanation: This class is responsible for the operations of the Student object such as delete, update and create.
Methods:
createEvent(event Event): This method is for creating an event.
deleteEvent(event Event): This method is for deleting an event.
editEvent(event Event): This method is for editing an event.
addParticipant(id int): This method adds a participant to an event.
deleteParticipant(id int): This method deletes a participant from an event.
updateAbout(about string): This method updates the about part of the event.
updateGePoints(gePoints int): This method is for updating student’s ge points.
updateDate(date Date): This method is for updating date.
updateCategory(category Category): This method is for updating category.
updateDuration(duration int): This method is for updating duration.
updateGeStatus(geStatus bool): This method is for updating GE status of the event.
updatePoster(poster Image): This method is for updating the poster.
addEvaluation(evaluation Evaluation): This method is for adding an evaluation.
addPossibleParticipant(possibleParticipant Student): This method is for adding a possible participant to the event.
addAttendedParticipant(attendedParticipant Student): This method updates the attend status of a student.
calculateAverageRate(): This method calculates the average rate.

3.5.2.6 BoardChairmanController Class

Explanation: This class is responsible for the operations of the BoardChairman object such as delete, update and create.
Methods:

demoteBoardMember(boardMember BoardMember): This method is for demoting a BoardMember to a Student.
promoteBoardMember(student Student): This method is for promoting a Student to a BoardMember.
createBoardChairman(): This method is for creating a Board Chairman.
deleteBoardChairman(): This method is for deleting a Board Chairman.

3.5.2.7 CampusMapController Class

Explanation: This class is responsible for the operations of the CampusMap object such as delete, update and create.
Methods:
getPlace(): This method returns the place.
editPlace(): This method is for editing the place.
showPin(place Place): This method shows the place in the campus map with a pin marker.
createCampusMap(): This method is for creating a campus map.
deleteCampusMap(): This method is for deleting a campus map.
updateCampusMap(): This method is for updating a campus map.

3.5.2.8 EvaluationController Class

Explanation: This class is responsible for the operations of the Evaluation object such as delete, update and create.
Methods:
createEvaluation(evaluation string, rate int): This method is for creating an evaluation with a comment and rate.
deleteEvaluation(evaluationId int): This method is for deleting evaluation.

3.5.2.9 ClubController Class

Explanation: This class is responsible for the operations of the Club object such as delete, update and create.
Methods:
createClub(): This method is for creating a club.
editClubInformation(club Club): This method is for editing the club information.
deleteClub(clubId int): This method is for deleting a club.
getClubs(): This method returns clubs.
addClubBoardMember(student Student): This method adds a board member to a club.
editClubCategory(club Club, category Category): This method is for editing the club category.
getClubLeaderBoard(): This method returns the club leaderboard.
calculateAverageRate(): This method calculates the club’s average rate.

3.5.2.9 LeaderBoardController Class

Explanation: This class is responsible for the operations of the LeaderBoard object such as delete, update and create.
Methods:
updateLeaderBoard(): This method is for updating the leaderboard.
deleteLeaderBoard(): This method is for deleting the leaderboard.
createLeaderBoard(): This method is for creating the leaderboard.

3.5.2.10 ClassroomController Class

Explanation: This class is responsible for the operations of the Classroom object such as delete, update and create.
Methods:

3.5.2.11 BuildingController Class

Explanation: This class is responsible for the operations of the Building object such as delete, update and create.
Methods:
createClassroom(): This method is for creating a classroom.
deleteClassroom(): This method is for deleting a classroom.
updateAvailable(): This method updates the availability.
updateCapacity(): This method updates the capacity.
updateReservedBlocks(): This method updates the reserved blocks.

3.5.2.12 PlaceController Class

Explanation: This class is responsible for the operations of the Place object such as delete, update and create.
Methods:
createPlace(): This method creates a place.
deletePlace(): This method deletes a place.
updatePlace(): This method updates a place.

3.5.2.13 OEMController Class

Explanation: This class is responsible for the operations of the OEM object such as delete, update and create.
Methods:
createOEM(): This method creates an OEM.
deleteOEM(): This method deletes an OEM.

3.5.3 Entity Classes

3.5.3.1 User Class

Explanation: This class is used for the users using the application such as students, club advisor, board members, board chairman and OEM.

Methods:
getFullName(): This method returns the full name.
setFullName(fullName string): This method sets the full name.
getEmail(): This method returns email.
setEmail(email string): This method sets an email.
encodePassword(): This method encodes the password.
decodePassword(): This method decodes the password.
getProfilePicture(): This method returns the profile picture.
setProfilePicture(profilePicture Image): This method sets a profile picture.

3.5.3.2 ClubAdvisor Class

Explanation: This class contains the management of club advisor objects and its operations. This class extends the User class.
Methods:
getClub(): This method returns the advisor's club.
setClub(): This method sets a club to the advisor.
getDepartment(): This method returns the department of the advisor.
setDepartment(department string): This method sets a department to the advisor.

3.5.3.3 Student Class

Explanation: This class contains the management of student objects and its operations. This class extends the User class.
Methods:
getDepartment(): This method returns the department of the student.
setDepartment(department string): This method sets the department of the student.
getHesCode(): This method returns the Hes code of the student.
getStudentID(): This method returns the student id of the student.
setStudentID(studentID string): This method sets the student ID of the student.
getGePoint(): This method returns the GE points of the student.
getFollowedClubs(): This method returns the clubs that are followed by the student.
getAttendedEvents(): This method returns the events that the student attended.
getAtteningEvents(): This method returns the events that the student will attend.
setGePoint(point int): This method sets the GE points of the student.

3.5.3.4 BoardMember Class

Explanation: This class contains the management of club board member objects and its operations. This class extends the Student class.
Methods:
getClub(): This method returns the club of board member.
setClub(club Club): This method sets a club to the board member.
createEvent(): This method is for creating an event.
cancelEvent(event Event): This method is for canceling an event.

updateEvent(event Event): This method is for updating an event.
getStatus(): This method returns the status of a board member.
setStatus(status Status): This method sets status to a board member.

3.5.3.5 BoardChairman Class

Explanation: This class contains the management of board chairman objects and its operations. This class extends the User class.
Methods:
getClub(): This method returns the club of the board chairman.
setClub(club Club): This method sets a club to a board chairman.

3.5.3.6 OEM Class

Explanation: This class contains the management of OEM objects and its operations. This class extends the User class.
Methods:
approveEvent(event Event): This method is for approving an event.
denyEvent(event Event): This method is for denying an event.

3.5.3.7 Event Class

Explanation: This class includes the fundamental operations of the events objects such as changing the time or getting rate.
Methods:
getName(): This method returns the name of the event.
setName(name string): This method sets the name of the event.
getCategory(): This method returns the category of the event.
setCategory(category Category): This method sets the category of the event.
getDate(): This method returns the date of the event.
setDate(date Date): This method sets the date of the event.
getDuration(): This method returns the duration of the event.
setDuration(duration int): This method sets the duration of the event.
getClub(): This method returns the club of the event.
setClub(club Club): This method sets the club of the event.
getCapacity(): This method returns the capacity of the event.
setCapacity(capacity int): This method sets the capacity of the event.
getAbout(): This method returns the about part of the event.
setAbout(about sting): This method sets the about part of the event.
getGeStatus(): This method returns the GE status of the event.
setGeStatus(geStatus int): This method sets the GE status of the event.
getGePoints(): This method returns the GE points of the event.
setGePoints(gePoints int): This method sets the GE points of the event.
getPoster(): This method returns the poster of the event.
setPoster(poster Image): This method sets the poster of the event.
getEvaluations(): This method returns the evaluations of the event.

getPossibleParticipants(): This method returns the possible participants of the event.
getAttendedParticipants(): This method returns the attended participants of the event.
getStatus(): This method returns the status of the event.
setStatus(status EventStatus): This method sets the rank of the event.
getRank(): This method returns the rank of the event.

3.5.3.8 Club Class

Explanation: This class includes the fundamental operations of the club objects such as changing the name, listing followers.
Methods:
getName(): This method returns the name of the club.
setName(name string): This method sets the name of the club.
getAbout(): This method returns the about part of the club.
setAbout(about string): This method sets the about part of the club.
getRate(): This method returns the rate of the club.
setRate(rate int): This method sets the rate of the club.
getCategory(): This method returns the category of the club.
setCategory(category Category): This method sets the category of the club.
getPastEvents(): This method returns the past events of the club.
setPastEvents(pastEvents Event[]): This method sets the past events of the club.
getUpcomigEvents(): This method returns the upcoming events of the club.
setUpcomigEvents(upcomigEvents Event[]): This method sets the upcoming events of the club.
getStudentsFollowed(): This method returns the students that follow the club.
setStudentsFollowed(studentsFollowed Student[]): This method sets the students that follow the club.
getBoardMembers(): This method returns the board members of the club.
getLogo(): This method returns the logo of the club.
setLogo(logo Image): This method sets the logo of the club.
getChairman(): This method returns the chairman of the club.
setChairman(cahirman BoardChairman): This method sets the chairman of the club.

3.5.3.9 Place Class

Explanation: This class is used for the place of the event objects and it includes building and classroom attributes.
Methods:
getName(): This method returns the name of the place.
setName(name string): This method set the name of the place.
getBuilding(): This method returns the building of the place.
setBuilding(building Building): This method sets the building of the place.

getClassroom(): This method returns the classroom of the place.
setClassroom(classroom Classroom): This method sets the classroom of the place.

3.5.3.10 Building Class

Explanation: This class contains the management of building objects and its operations. This class extends the Place class.
Methods:
getName(): This method returns the name of the building.
setName(name string): This method sets the name of the building.
getFaculty(): This method returns the faculty of the building.
setFaculty(faculty string): This method sets the faculty of the building.
getIsAvailable(): This method returns the availability of the building.
setIsAvailable(isAvailable bool): This method sets the availability of the building.
getClassrooms(): This method returns the classrooms of the building.
setClassrooms(classrooms Classroom[]): This method sets the classrooms of the building.

3.5.3.11 Classroom Class

Explanation: This class contains the management of classroom objects and its operations. This class extends the Place class.
Methods:
getName(): This method returns the name of the classroom.
setName(name string): This method sets the name of the classroom.
getReservedBlocks(): This method returns the reserved blocks of the classroom.
getCapacity(): This method returns the capacity of the classroom.
getIsAvailable(): This method returns the availability of the classroom.

3.5.3.12 CampusMap Class

Explanation: This class is used for the campus map. It includes the place of buildings.
Methods:
getBuildings(): This method returns the buildings of the campus map.
setBuildings(buildings Building[]): This method sets the buildings of the campus map.
getMapImage(): This method returns the map image of the campus map.
setMapImage(mapImage Image): This method sets the map image of the campus map.

3.5.3.13 Image Class

Explanation: This class is used for the users' and events' profile photographs. This class extends the Python image class.
Methods:

getPath(): This method returns the path of the image.
setPath(path string): This method sets the path of the image.
getWidth(): This method returns the width of the image.
setWidth(width int): This method sets the width of the image.
getHeight(): This method returns the height of the image.
setHeight(height int): This method sets the height of the image.
getFormat(): This method returns the format of the image.
setFormat(format string): This method sets the format of the image.

3.5.3.14 Evaluation Class

Explanation: This class is used for the evaluation of the events. It stores the comments and rates of the event made by the users.
Methods:
getComment(): This method returns the comment of the evaluation.
setComment(comment string): This method sets the comment of the evaluation.
getRate(): This method returns the rate of the evaluation.
setRate(rate int): This method sets the rate of the evaluation.
getOwner(): This method returns the owner of the evaluation.
setOwner(owner Student): This method sets the owner of the evaluation.
getEvent(): This method returns the event of the evaluation.
setEvent(event Event): This method sets the event of the evaluation.

3.5.3.15 PSIScore Class

Explanation: This class is used for the Personal Sociability Intelligence score of the users. The object of this class is affected by the activities of the user such as participating in events, following clubs and so on.
Methods:
calculateScore(): This method calculates the score.
resetScore(): This method resets the score.

4. Improvement Summary

There are some improvements in the second iteration over the first iteration. These improvements are briefly described below.

4.1 General

- Figure numbers have been added to each figure.

4.2 Subsystem Decomposition

- Each subsystem in each layer has been explained in text.

4.3 Deployment Diagram

- Deployment diagram has been added to the report.
- Explanation of deployment diagram has been added below the deployment diagram.

4.4 Hardware/Software Mapping

- The system requirements of the production server have been added.

4.5 Persistent Data Management

- Classes to be stored in the database have been specified.

4.6 Access Control and Security

- Access matrix has been added to the report.
- The access control matrix consists of actors, classes and access rights.
- Access right lists the operations that can be executed on instances of the class by the actor.

4.7 Boundary Conditions

- It has been added what the admin needs to do to initialize the application on the development server to the initialization section.
- It has been added what the admin needs to do to terminate the application on the development server to the termination section.

4.8 Final Object Design

- Structure of design patterns and some classes have been added to abstraction of the final object design.
- Structure of design patterns and some classes have been added to classes of boundary objects.
- Structure of design patterns and some classes have been added to classes of controller objects.
- Structure of design patterns and some classes have been added to classes of entity objects.

4.9 Class Interfaces

- Explanations and methods of the controller and entity classes have been added.
- Some methods of the boundary class have been changed.
- Multiplicities have been reviewed.

5. Glossary & References

5.1 Glossary

PSI: This term is the abbreviation of Personal Sociability Intelligence. This score indicates how social a person is. This score will be calculated by an algorithm determined by the system. The events that the user participates in, the clubs the user follows, the clubs the user is the board member of will affect the calculation of this score.

5.2 References

[1] “Documentation - Chrome Developers”. https://developer.chrome.com/docs/. [Accessed: Nov 24, 2021].

[2] “Microsoft Edge Documentation”. https://docs.microsoft.com/en-us/microsoft-edge/. [Accessed: Nov 24, 2021].

[3] “Firefox - Mozilla”. https://docs.microsoft.com/en-us/microsoft-edge/. [Accessed: Nov 24, 2021].

[4] “Opera Version History”. https://help.opera.com/en/opera-version-history/. [Accessed: Nov 24, 2021].

[5] “Resources - Safari - Apple Developers”. https://developer.apple.com/safari/resources/. [Accessed: Nov 24, 2021].

[6] “Django”. https://www.djangoproject.com/. [Accessed: Nov 24, 2021].

[7] “The Progressive JavaScript Framework”. https://vuejs.org/. [Accessed: Nov 24, 2021].

[8] “PostgreSQL: The World's Most Advanced Open Source Relational Database”. https://www.postgresql.org/. [Accessed: Nov 24, 2021].

[9] “AWS Documentation”. https://docs.aws.amazon.com/. [Accessed: Nov 24, 2021].