CS 319 Object-Oriented Software Engineering

Instructor: Eray Tüzün, Teaching Assistant: Yahya Elnouby

AdayBilgi

Deliverable 1 - 2nd Iteration

Group 1

Emine Noor

Eray İşçi

Hatice Kübra Çağlar

İbrahim Çaycı

İrem Damla Karagöz

Yiğit Özhan

Table of Contents

1. Use Case Diagram

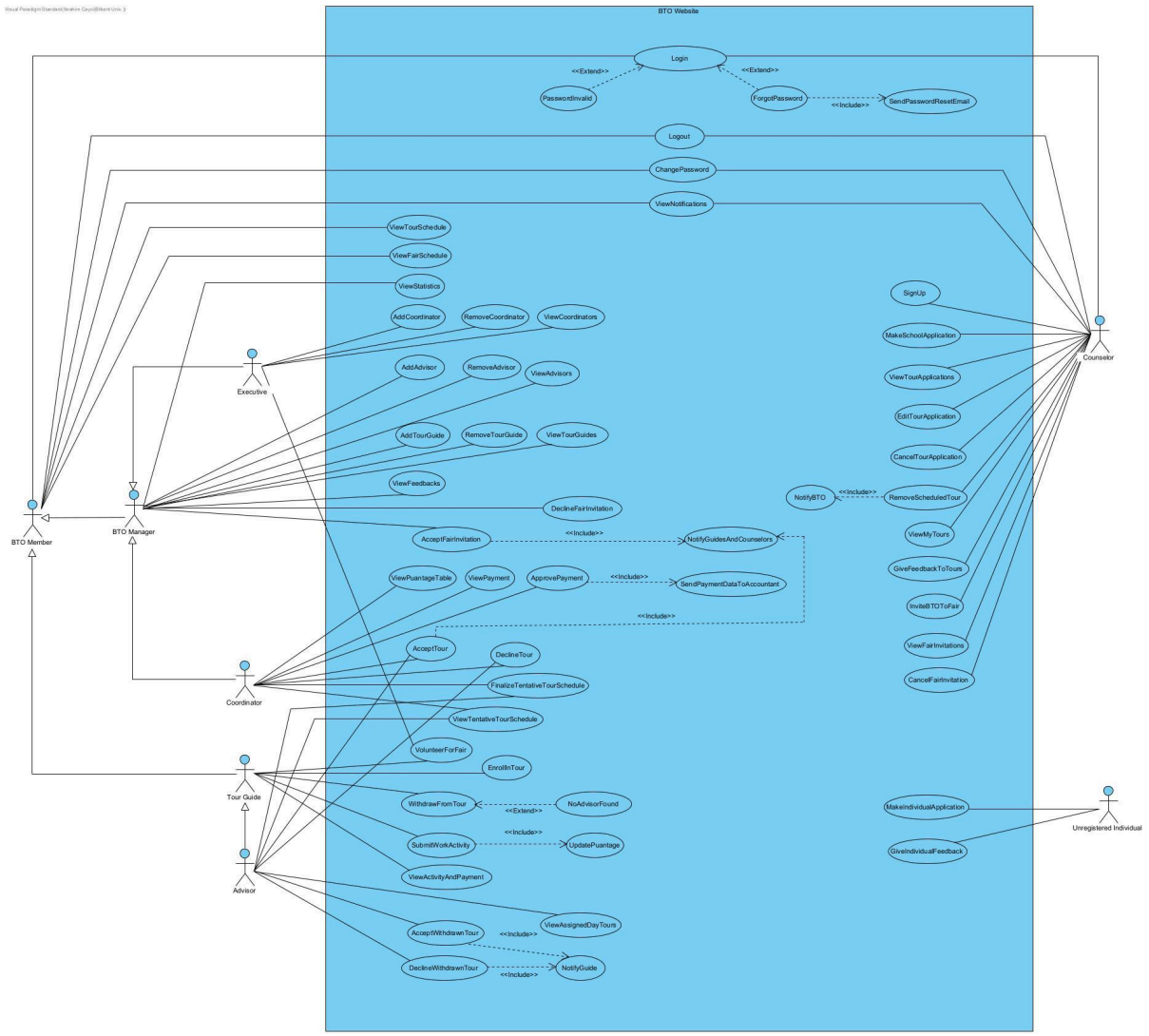

Figure 1: Use Case Diagram of BTO Website

2. Textual Use Case Description

1. **Name:** SignUp
2. **Participating Actor:** Counselors (Other users' accounts are assigned automatically by the system)
3. **Entry Condition:** Counselor requests to create an account, and navigates to the signup page.
4. **Exit Condition:** Account is created.
5. **Flow of Events:**
    5.1. Counselor provides the necessary information, such as his name, surname, phone number, highschool, to open an account.
    5.2. System checks account credentials and if the high school credentials exist.
    5.3. If the credentials are correct, the system logs the user into the account created.
    5.4. System navigates the counselor to the dashboard page for the counselor.
6. **Quality Requirements:** None

1. **Name:** Login
2. **Participating Actor:** BTO Members, Counselors
3. **Entry Condition:** User requests to log in, and navigates to the login page.
4. **Exit Condition:** User successfully logs in.
5. **Flow of Events:**
    5.1. User provides their email, and current password.
    5.2. System checks the user's credentials.
    5.3. User is logged in if the credentials are valid.
    5.4. System navigates the counselor to the dashboard page.
6. **Quality Requirements:** None

1. **Name:** PasswordIsInvalid
2. **Participating Actor:** BTO Members, Counselors
3. **Entry Condition:** User attempts to log in with an incorrect password.
4. **Exit Condition:** System displays an error message indicating that the password is invalid.
5. **Flow of Events:**
    5.1. User enters a password during login.

5.2. System checks the credentials and detects that the password is incorrect.

5.3. System displays an error message and prompts the user to try again.

6. **Quality Requirements:** None

1. **Name:** ForgotPassword
2. **Participating Actor:** BTO Members, Counselors
3. **Entry Condition:** User forgets his password, and clicks “Forgot password?” button on the login page.
4. **Exit Condition:** User successfully logs in.
5. **Flow of Events:**

   5.1. User enters his email to get a link to reset the password.

   5.2. System sends an email containing a link to reset the password.

   5.3. System notifies the user that the email is sent.

   5.4. User resets password following the link.

   5.5. System redirects the user back to the login page.

6. **Quality Requirements:** None

1. **Name:** SendPasswordResetEmail
2. **Participating Actor:** BTO Members, Counselors
3. **Entry Condition:** User clicks “Forgot password?” button on the login page.
4. **Exit Condition:** An email containing a link to reset password is successfully sent.
5. **Flow of Events:**

   5.1. System verifies that the submitted email address is associated with a registered account.

   5.2. System generates a unique link for resetting password.

   5.3. System sends the email to the user’s email address.

6. **Quality Requirements:** Notifications should be sent within 3 seconds.

1. **Name**: Logout
2. **Participating Actor**: BTO Members, Counselors
3. **Entry Condition**: User selects the option to log out.
4. **Exit Condition**: User is successfully logged out, and the system returns to the main screen.
5. **Flow of Events**:

5.1. System receives the logout request, and confirms the action.

5.2. System redirects the user to the main screen.

6. **Quality Requirements:** Modifications must be synchronized within 3 seconds.

1. **Name:** ChangePassword
2. **Participating Actor:** BTO Members, Counselors
3. **Entry Condition:** User navigates to the change password option.
4. **Exit Condition:** User's password is successfully changed.
5. **Flow of Events:**

   5.1. User enters their current password, new password, and confirms the new password.

   5.2. System validates the current password.

   5.3. If the current password is correct and the new password matches the confirmation, the password is updated in the system.

   5.4. System notifies the user that the password has been successfully changed.

6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds.

1. **Name:** ViewTourSchedule
2. **Participating Actor:** BTO Members
3. **Entry Condition:** BTO Member requests to view the tour schedule, and navigates the tour schedule page.
4. **Exit Condition:** The system displays the schedule.
5. **Flow of Events:**

   5.1. System retrieves the tour schedule from the database.

   5.2. The schedule is displayed to the BTO Member.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: ViewFairSchedule
2. **Participating Actor**: BTO Members
3. **Entry Condition**: BTO Member requests to view the schedule for upcoming fairs (either accepted, declined or pending).
4. **Exit Condition**: Fair schedule is displayed to the BTO Member.

5. **Flow of Events**:
    5.1. BTO Member selects the fair schedule button.
    5.2. System retrieves the fair schedule from the database.
    5.3. System displays the fairs and related information.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: ViewStatistics
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manger requests to view tour and guide statistics, navigates to the statistics page.
4. **Exit Condition**: Statistical data is displayed to the BTO Manager.
5. **Flow of Events**:
    5.1. System retrieves the tour and guide statistics from the database.
    5.2. System displays relevant statistics.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: AddCoordinator
2. **Participating Actor**: Executives
3. **Entry Condition**: Executive decides to add a new coordinator to the system, navigates to the add coordinator management section.
4. **Exit Condition**: Coordinator is successfully added to the system.
5. **Flow of Events**:
    5.1. Executive selects a tour guide, or advisor to change his role as a coordinator.
    5.2. System saves the new coordinator details.
    5.3. System confirms the successful addition of the coordinator.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds

1. **Name**: RemoveCoordinator
2. **Participating Actor**: Executives
3. **Entry Condition**: Executive wants to remove an existing coordinator from the system, and navigates to the coordinator management section.
4. **Exit Condition**: Coordinator is successfully removed from the system.
5. **Flow of Events**:
    5.1. Executive selects the coordinator they wish to remove and clicks the remove button.
    5.2. System prompts for confirmation of the removal.
    5.3. System deletes the coordinator's details.
    5.4. System updates relevant records and confirms the removal.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds.

1. **Name**: ViewCoordinators
2. **Participating Actor**: Executives
3. **Entry Condition**: Executive requests to view the current list of coordinators, and navigates to the coordinator management section.
4. **Exit Condition**: System displays the list of all coordinators.
5. **Flow of Events**:
    5.1. Executive selects the option to view the list of coordinators.
    5.2. System retrieves coordinator list from the database.
    5.3. System displays the list with coordinator details.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: AddAdvisor
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager decides to add a new advisor to the system, and navigates to the add advisor management section.
4. **Exit Condition**: Advisor is successfully added to the system.
5. **Flow of Events**:
    5.1. BTO Manager selects the option to add an advisor.
    5.2. BTO Manager selects a tour guide to change his role as an advisor.
    5.3. System updates the new advisor details.

5.4. System confirms the successful addition of the advisor.

6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds.

1. **Name**: RemoveAdvisor
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: A BTO Manager decides to remove an existing advisor, and navigates to the advisor management section.
4. **Exit Condition**: The advisor is successfully removed from the system.
5. **Flow of Events**:
   5.1. BTO Manager selects the option to remove an advisor.
   5.2. BTO Manager selects the advisor they wish to remove and clicks the remove button.
   5.3. System prompts for confirmation of the removal.
   5.4. System changes the member's details from "advisor" to "tour guide".
   5.5. System updates relevant records and confirms the removal.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds.

1. **Name**: ViewAdvisors
2. **Participating Actor**: BTO Members
3. **Entry Condition**: A BTO Member requests to view the current list of advisors.
4. **Exit Condition**: The system displays the list of advisors.
5. **Flow of Events**:
   a. The BTO Member navigates to the advisor management section.
   b. The BTO Member selects the option to view the list of advisors.
   c. The system retrieves and displays the list with advisor details.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: AddTourGuides
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to add a new tour guide.
4. **Exit Condition**: New tour guide is added to the system.

5. **Flow of Events**:
    5.1. BTO Manager navigates to the guides list page.
    5.2. System displays the list of all registered tour guides.
    5.3. BTO Member enters the guide's details.
    5.4. System saves the new guide information and updates the guides list.
    5.5. System displays the updated guides list.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds.

1. **Name**: RemoveTourGuides
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to remove an existing tour guide.
4. **Exit Condition**: Tour guide is removed from the system.
5. **Flow of Events**:
    5.1. BTO Manager navigates to the guides list page.
    5.2. System displays the list of all registered tour guides.
    5.3. BTO Manager selects the guide to remove.
    5.4. System removes the guide from the list and updates records.
    5.5. System displays the updated guides list.
6. **Quality Requirements:** None

1. **Name**: ViewTourGuides
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to view the list of tour guides.
4. **Exit Condition**: Tour guides list is displayed to the BTO Manager.
5. **Flow of Events**:
    5.1. BTO Manager navigates to the guides list page.
    5.2. System retrieves guide list from the database.
    5.3. System displays the list of the guides with details.
6. **Quality Requirements:** None

1. **Name**: ViewPuantageTable
2. **Participating Actor**: Coordinator
3. **Entry Condition**: Coordinator requests to view the puantage table for tour guides and advisors, and navigates to the puantage table page.
4. **Exit Condition**: Puantage table is displayed.
5. **Flow of Events**:
   5.1. System retrieves data.
   5.2. System displays monthly puantage table details for tour guides and advisors.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: ApprovePayment
2. **Participating Actor**: Coordinators
3. **Entry Condition**: Coordinator requests to approve payment details for tour guides and advisors, and navigates to the payment details page.
4. **Exit Condition**: Payment details are approved by the coordinator.
5. **Flow of Events**:
   5.1. System displays payment details for tour guides and advisors.
   5.2. Coordinator approves payment details.
   5.3. System sends the payment data to the accountant.
6. **Quality Requirements:** All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: SendPaymentDataToAccountant
2. **Participating Actor**: Coordinators
3. **Entry Condition**: Coordinator approves payment details.
4. **Exit Condition**: Payment details are sent to the accountant email addresses.
5. **Flow of Events**:
   5.1. System creates an email with payment details.
   5.2. System sends the payment email to the accountant.
6. **Quality Requirements:** All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewPayment
2. **Participating Actor**: Coordinators
3. **Entry Condition**: Coordinator requests to view payment details for tour guides and advisors.
4. **Exit Condition**: Payment details are displayed to the coordinator.
5. **Flow of Events**:
    5.1. Coordinator navigates to the payment details page.
    5.2. System displays payment details for tour guides and advisors.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: AcceptTour
2. **Participating Actor**: Coordinators, Advisors
3. **Entry Condition**: User requests to accept a certain tour assigned to a time slot by the system.
4. **Exit Condition**: The tour is accepted with current details (assigned guide, time slot etc.).
5. **Flow of Events**:
    5.1. User navigates to the tour schedule.
    5.2. User clicks on a certain tour and clicks accept.
    5.3. The tour is updated as accepted.
    5.4. The tour guide that volunteered for the tour is notified.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: DeclineTour
2. **Participating Actor**: Coordinators, Advisors
3. **Entry Condition**: User requests to decline a certain tour assigned to a time slot by the system.
4. **Exit Condition**: The tour is declined with current details (assigned guide, time slot etc.).
5. **Flow of Events**:
    5.1. Usernavigates to the tour schedule.
    5.2. User clicks on a certain tour and clicks decline.
    5.3. The tour is updated as declined.
    5.4. The tour guide that volunteered for the tour is notified about the update.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewTentativeTourSchedule
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to view the schedule created as a tentative by the system for upcoming tours, and navigates to the tour schedule page.
4. **Exit Condition**: Tentative tour schedule is displayed.
5. **Flow of Events**:
   5.1. System retrieves schedule from database.
   5.2. System displays the tentative tour schedule.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: FinalizeTentativeTourSchedule
2. **Participating Actor**: Coordinators, Advisors
3. **Entry Condition**: User requests to finalize the schedule created as a tentative by the system for upcoming tours, and navigates to the tour schedule page.
4. **Exit Condition**: Tentative tour schedule is finalized and saved in the system.
5. **Flow of Events**:
   5.1. System displays the tentative tour schedule.
   5.2. User confirms the finalized schedule.
   5.3. System notifies BTO Members about schedule confirmation.
   5.4. System displays the updated tour schedule.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: AcceptFairInvitations
2. **Participating Actor**: Coordinators, Advisors
3. **Entry Condition**: User requests to accept a fair invitation of highschool, and navigates to the fair invitations page.
4. **Exit Condition**: Fair is accepted and saved in the system.

5. **Flow of Events**:
    5.1. System retrieves and displays the fair invitations.
    5.2. User accepts the fair invitation.
    5.3. System notifies high school counselors and BTO Members about the update.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: DeclineFairInvitations
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to decline a fair invitation of highschool, and navigates to the fair invitations page.
4. **Exit Condition**: Fair is declined and saved in the system.
5. **Flow of Events**:
    5.1. System retrieves and displays the fair invitations.
    5.2. BTO Manager declines the fair invitation.
    5.3. System notifies high school counselor and BTO Members about the update.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: NotifyGuidesAndCounselors
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager accepts tour or fair invitation with assigned guides.
4. **Exit Condition**: Tour guides enrolled in accepted tours or volunteered for fair invitations are notified. Counselor of the tour is notified.
5. **Flow of Events**:
    5.1. System identifies the tour guides assigned to the accepted tour or those who volunteered for the fair invitation.
    5.2. System prepares a notification message with details about the accepted tour or fair invitation.
    5.3. System sends the notification to each assigned or volunteered tour guide, and counselor of the tour.
6. **Quality Requirements:** Notifications should be sent and received within 3 seconds. Data retrieval from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewFeedbacks
2. **Participating Actor**: BTO Managers
3. **Entry Condition**: BTO Manager requests to view feedback on past tours, and navigates to the feedback page.
4. **Exit Condition**: Feedback details are displayed to the BTO Manager.
5. **Flow of Events**:
    5.1. System retrieves feedbacks from the database.
    5.2. System displays feedbacks given by counselors for past tours and individuals for past individual tours.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name:** EnrollInTour
2. **Participating Actor:** Tour Guides
3. **Entry Condition:** Tour Guide requests to enroll in a tour.
4. **Exit Condition:** Tour Guide completes the enrollment process.
5. **Flow of Events:**
    5.1. Tour Guide selects a tour to enroll in.
    5.2. System checks tour availability and enrollment criteria (one guide per 60 visitors).
    5.3. If the tour is not already assigned to enough guides, the system accepts the guide's enrollment.
    5.4. The system confirms the enrollment and updates the Tour Guide's schedule.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name:** WithdrawFromTour
2. **Participating Actor:** Tour Guides
3. **Entry Condition:** Tour Guide has enrolled in a tour, and wants to withdraw, and navigates to his scheduled tours.
4. **Exit Condition:** Tour Guide successfully withdraws the tour.
5. **Flow of Events:**

5.1. Tour Guide selects the option to withdraw.

5.2. System appoints an advisor or a tour guide for the tour.

5.3. System confirms the withdrawal request and the tour guide is withdrawn from the tour.

5.4. System updates the tour guide's schedule.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: NoAdvisorFound
2. **Participating Actor**: Tour Guide
3. **Entry Condition**: Tour Guide requests to withdraw from a tour that he accepted before.
4. **Exit Condition**: System notifies Tour Guide that no advisor is available, and the withdrawal request is denied.
5. **Flow of Events**:

5.1. Tour Guide requests withdrawal process for an assigned tour.

5.2. System checks if an advisor is available for the tour.

5.3. System finds that no advisor is available for the tour.

5.4. System displays an error message notifying the Tour Guide that they cannot withdraw from the tour due to the lack of an available advisor.

6. **Quality Requirements:** Data retrieval from the database for a page should take less than 1 second. Modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: AcceptWithdrawnTour
2. **Participating Actor**: Advisor
3. **Entry Condition**: A tour guide has submitted a request to withdraw from a tour, and the request is pending advisor approval.
4. **Exit Condition**: Advisor accepts withdrawal request, and the Tour Guide is officially removed from the tour.
5. **Flow of Events**:

5.1. Advisor reviews the details of the withdrawal request of the tour guide.

5.2. Advisor accepts to approve the withdrawal.

5.3. System processes the approval and removes the tour guide from the tour.

5.4. System sends a notification to the Tour Guide confirming that their withdrawal request has been accepted.

6. **Quality Requirements:** Modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: DeclineWithdrawnTour
2. **Participating Actor**: Advisor
3. **Entry Condition**: A tour guide has submitted a request to withdraw from a tour, and the request is pending advisor approval.
4. **Exit Condition**: Advisor declines withdrawal request, and the Tour Guide is officially removed from the tour.
5. **Flow of Events**:
   5.1. Advisor reviews the details of the withdrawal request of the tour guide.
   5.2. Advisor declines to approve the withdrawal.
   5.3. System sends a notification to the Tour Guide confirming that their withdrawal request has been accepted.
6. **Quality Requirements:** Modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: NotifyGuide
2. **Participating Actor**: Advisor
3. **Entry Condition**: Advisor accepts or declines a withdrawal request submitted by a Tour Guide for an assigned tour.
4. **Exit Condition**: Tour Guide is notified of the advisor's decision regarding their withdrawal request.
5. **Flow of Events**:
   5.1. System detects the advisor's decision and initiates the notification process.
   5.2. System generates a notification message indicating whether the withdrawal request has been accepted or declined.
   5.3. System sends the notification to the Tour Guide.
6. **Quality Requirements:** Notifications should be sent and received within 3 seconds.

1. **Name**: ViewAssignedDayTours
2. **Participating Actor**: Advisor
3. **Entry Condition**: Advisor requests to view the list of tours of the day when advisor is assigned.
4. **Exit Condition**: The system displays a list of all day tours assigned to the Advisor.
5. **Flow of Events**:
    5.1. System retrieves a list of all upcoming day tours assigned to the Advisor's advisees.
    5.2. System displays the list of assigned day tours.
6. **Quality Requirements:** Data retrieval from the database for a page should take less than 1 second.

1. **Name:** VolunteerForFair
2. **Participating Actor:** Tour Guides, Executives
3. **Entry Condition:** User wants to volunteer for a fair.
4. **Exit Condition:** User is successfully assigned to fair.
5. **Flow of Events:**
    5.1. User selects the fair event they wish to volunteer for.
    5.2. System checks if there are available volunteer slots.
    5.3. If slots are available, the User is added as a volunteer.
    5.4. The system confirms the User's registration as a volunteer.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name:** SubmitWorkActivity
2. **Participating Actor:** Tour Guides
3. **Entry Condition:** Tour Guide is logged in and wants to submit extra work activity (like last minute tour assignment, or other activity).
4. **Exit Condition:** The work activity is successfully submitted.
5. **Flow of Events:**
    5.1. Tour Guide navigates to the work activity page.
    5.2. Tour Guide fills out details of the work activity, such as category, title, date and time.
    5.3. System validates and records the submission.
    5.4. The system confirms that the activity has been successfully submitted.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewActivityAndPayment
2. **Participating Actor**: Tour Guides
3. **Entry Condition**: Tour Guide requests to view his/her work activity, navigates to the work activity page.
4. **Exit Condition**: Work activity details are displayed to the Tour Guide.
5. **Flow of Events**:
   5.1. System retrieves guide list from the database.
   5.2. System displays the work activity.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: MakeSchoolApplication
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor decides to submit a new school tour application, and navigates to the school application page.
4. **Exit Condition**: The school tour application is successfully submitted.
5. **Flow of Events**:
   5.1. Counselor chooses a day and time for the tour from the application page.
   5.2. If all time and day are chosen, the system accepts and submits the application.
   5.3. System displays a confirmation message indicating the successful tour application submission.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewTourApplications
2. **Participating Actor:** Counselors
3. **Entry Condition:** Counselor wants to view their applications.
4. **Exit Condition:** The system displays the selected tour application details.
5. **Flow of Events:**

5.1. User selects a tour application from their account.

5.2. System retrieves the current data of the application.

5.3. The data is displayed to the user (e.g., status, selected dates etc.).

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name:** EditTourApplication
2. **Participating Actor:** Counselors
3. **Entry Condition:** Counselor requests to edit an existing tour application which is pending.
4. **Exit Condition:** The tour application is successfully updated.
5. **Flow of Events:**

5.1. Counselor navigates to their submitted tour applications.

5.2. Counselor selects the option to edit the application.

5.3. Counselor changes some information about the tour application, such as time and date, and submits them.

5.4. The system validates and updates the application.

5.5. The system confirms the successful update.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name:** CancelTourApplication
2. **Participating Actor:** Counselors
3. **Entry Condition:** Counselor requests to cancel an existing tour application.
4. **Exit Condition:** The tour application is successfully canceled.
5. **Flow of Events:**

5.1. Counselor navigates to their submitted tour applications.

5.2. Counselor selects the option to cancel an application.

5.3. System confirms the cancellation.

5.4. The system removes the application and updates the tour availability.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second. All data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: RemoveScheduledTour
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor requests to remove a scheduled tour from the system, and navigates to my tours page.
4. **Exit Condition**: The scheduled tour is removed and updated in the system.
5. **Flow of Events**:
    5.1. Counselor selects the scheduled tour to remove.
    5.2. System removes the tour from the schedule and updates relevant records.
    5.3. System notifies BTO members.
    5.4. System displays the updated tour list of Counselor.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: NotifyBTO
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor removes a scheduled tour from the system.
4. **Exit Condition**: BTO is notified about the tour removal action.
5. **Flow of Events**:
    5.1. Counselor removes a tour.
    5.2. System notifies BTO members.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.** Notifications should be sent and received within 3 seconds.

1. **Name:** ViewMyTours
2. **Participating Actor:** Counselors
3. **Entry Condition:** Counselor requests to view all available tours, and navigates to the tours page.
4. **Exit Condition:** The system displays a list of scheduled tours.
5. **Flow of Events:**
    5.1. System retrieves and displays the list of tours.
    5.2. Counselor can view details for each tour if desired.

6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: GiveFeedbackToTours
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor completes a tour and wants to provide feedback, and navigates to the feedback page.
4. **Exit Condition**: Feedback is saved in the system.
5. **Flow of Events**:
    5.1. Counselor enters a comment and a rating as a feedback and submits.
    5.2. System saves the feedback for review.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewNotifications
2. **Participating Actor**: Counselors, BTO Members
3. **Entry Condition**: User accesses their notification list.
4. **Exit Condition**: Notifications are displayed to the user.
5. **Flow of Events**:
    5.1. User navigates to the notifications page.
    5.2. System displays the latest notifications.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: InviteBTOToFair
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor requests to invite BTO to upcoming fairs, and navigates to the fair invitation page.
4. **Exit Condition**: Fair invitation is successfully submitted.
5. **Flow of Events**:
    5.1. Counselor selects date and time for a fair invitation and submits it.
    5.2. System saves the fair invitation.

6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: ViewFairInvitations
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor requests to see their previous , and navigates to the fair invitation page.
4. **Exit Condition**: Fair invitations are displayed.
5. **Flow of Events**:
   5.1. System retrieves previous fair invitations from database.
   5.2. System displays previous fair invitations of counselor.
6. **Quality Requirements:** Data retrieval and display from the database for a page should take less than 1 second.

1. **Name**: CancelFairInvitation
2. **Participating Actor**: Counselors
3. **Entry Condition**: Counselor requests to cancel fair invitation, and navigates to the fair invitation page.
4. **Exit Condition**: Fair invitation is successfully canceled.
5. **Flow of Events**:
   5.1. Counselor selects a fair invitation and cancels it.
   5.2. System deletes the fair invitation.
6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: MakeIndividualApplication
2. **Participating Actor**: Unregistered Individuals
3. **Entry Condition**: Unregistered Individual wants to apply individually for a tour, and navigates to the individual tour application page.
4. **Exit Condition**: Application is submitted and saved in the system.
5. **Flow of Events**:

5.1. Unregistered Individual fills out the form in the individual tour application page by entering his name, surname, highschool, phone number, email, tour date and visitor count and submits it.

5.2. System saves the application.

6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

1. **Name**: GiveIndividualFeedback
2. **Participating Actor**: Unregistered Individuals
3. **Entry Condition**: Unregistered Individual has completed a tour, and receives an email with a link to “Send feedback” page.
4. **Exit Condition**: Individual feedback is saved in the system.
5. **Flow of Events**:

5.1. Unregistered Individual clicks the link.

5.2. Unregistered Individual enters a comment and a rating as a feedback and submits it.

5.3. System saves the feedback for review.

6. **Quality Requirements:** Data updates or modifications must be stored and synchronized within 3 seconds**.**

3. Tech Stack

3.1 Backend

We decided to use the **Spring Boot** framework of Java for backend purposes.

- It provides built-in tools for creating RESTful APIs quickly, which is ideal for developing the business logic that our application requires, such as managing tours, users, schedules, and feedback.
- Since Spring Boot provides security with built-in tools such as **Spring Security,** we can implement strong authentication, authorization, and other security mechanisms efficiently. We have sensitive data such as High School prioritization, feedback mechanisms, and payment control. These functionalities should be protected with a strong and secure system, which Spring Boot provides.
- Spring Boot easily integrates with relational databases using Spring Data JPA. This feature allows us to do CRUD (Create, Read, Update, Delete) operations easily.
- It is a widely used framework with a large community to help us resolve our problems.

3.2 Frontend

We decided to use React, a Javascript library, in addition to HTML/CSS for frontend purposes.

- React is designed to work seamlessly with RESTful APIs, which fits perfectly with the Spring Boot backend.
- The React ecosystem offers a wide range of libraries and tools, such as React Router for handling navigation and Redux for state management. These are useful for building complex interfaces like role-based dashboards (e.g., different views for coordinators, counselors, and executives).
- React uses a virtual DOM, which optimizes rendering and updates, ensuring the front end is fast and responsive. This is particularly important for applications with real-time updates, such as notifications about tour statuses.

3.3 Database Management

We decided to use a relational database management system called MySQL for database operations.

- MySQL integrates well with Spring Boot using Spring Data JPA. This allows us to work with the database using Java entities.
- It ensures secure data storage. This ensures that sensitive information, such as user credentials and tour data, remains secure.

- MySQL is open-source, making it a cost-effective solution with enterprise-grade features. It has a large community and extensive documentation, providing support and guidance for best practices.