CS319 Object-Oriented Software Engineering
Project Deliverable 1 (D1) - S2T7 - agora

Team Members:

- Bertan Uran - 22102541
- Egehan Yıldız - 22203014
- Ekin Köylü - 22103867
- Emre Yazıcıoğlu - 22201668
- İlke Latifoğlu - 22203818
- Merve Güleç - 22103231

Contents of the Document

Introduction

The Bilkent Information Office (BTO) project is an organization that undertakes the promotion of Bilkent University and carries out this through campus tours and fairs, and the project aims to digitize and automate this process. The current system uses Google Forms for applications and Excel spreadsheets for managing the schedules. As the system goes over manually, this leads to some potential errors and waste of time.

Our project aims to develop an automated system that enables high school prioritization, campus tours, and fairs, and facilitates easier communication between guides. The system will include some features to facilitate the work of the officers in the Information Office throughout the process. It includes basic processes such as tour applications made by high schools, tour approvals made by consultants, tours by guides, feedback collection system in different forms received from actors such as candidate guides, high schools, and guides, tables containing information about guides, payroll tracking for the table for tracking the scoring system. The system will provide viewing and editing access to different responsibilities of different users with customized access and functionality based on each user's responsibilities.

Non-functional Requirements for the Project

User Interface (UI) and Usability

1. **Intuitive Navigation**

- The system must offer a well-organized, intuitive navigation structure that allows users (coordinators, advisors, guides, candidates) to easily access the functionalities they need within 4 clicks or less.
    - ***Note:*** *Hereafter, the term* ***"a coordinator"*** *will refer to the coordinator, secretary, or director within the Bilkent Information Office. Due to the structure of the application, it has been decided that all of them will share the same access level.*
- For example, an advisor should be able to log in, navigate to "Highschool Application Tracking System" and select a specific tour to confirm it, all within three clicks from the dashboard.

2. **Responsive Design**

- The platform should be fully responsive, meaning it must work seamlessly across various devices, including desktop computers, tablets, and smartphones. The layout must adapt fluidly to different screen sizes, ensuring the user experience is consistent and user-friendly regardless of the device.

3. **Accessibility**

- The system must adhere to ***WCAG 2.1 AA standards*** to ensure accessibility for users with disabilities. For example, keyboard navigation should allow users to complete tasks such as form submissions without using a mouse, and screen readers should accurately describe buttons like "Submit Feedback".

4. **User Guidance and Error Handling**

- For workflows that might be complex (like submitting feedback or scheduling tours), the system should provide clear guidance. This includes tooltips, in-line instructions, and easy access to user help documentation within 2 clicks.
- For instance, when submitting feedback, a tooltip could explain how to fill out each field, and if an error occurs, the system should prompt users with specific corrective instructions, like "Use MM/DD/YYYY for the date".

5. **Usability**

- The platform should be easy and intuitive for all users, including non-technical users like guides and candidates. Usability testing should ensure that tasks can be completed efficiently with minimal effort, and users should be able to complete their core tasks without needing extensive help or training.

- For example, a guide should be able to log their tour hours in under 30 seconds by selecting the tour, entering the duration, and submitting—all from a single, easy-to-use page (Information Office calls this page the “Puantaj Page”).

6. **Customizability**

- The platform should allow users to customize their interface based on their preferences. For example, users should be able to choose between light and dark modes, rearrange dashboard widgets, or configure notification settings based on their role and needs.

Security, Privacy, and Compliance

1. **Role-Based Access Control (RBAC)**

- The system will implement strict access control where users only see data relevant to their roles. For example, a coordinator can view and edit all tour and financial data, while a guide only has access to basic tour details and cannot view sensitive information like IBAN numbers.

2. **Logging and Auditing**

- All important actions, such as editing high school application forms or tour feedback, must be logged with information about who made the changes and when. For instance, if an advisor modifies tour details, this should be recorded in the log, which is accessible only to authorized personnel.

3. **Privacy Measures**

- The system should comply with **GDPR** and **KVKK** regulations, ensuring users can manage their data. For instance, if a guide requests their data to be deleted, the system should process this request *within 24 hours*, in compliance with data protection laws.

4. **Safety and Dependability**

- The system should prevent any actions that could lead to system malfunction or user harm. For instance, it should block unauthorized access attempts and handle errors in a way that doesn’t compromise system integrity or user trust.

Performance and Efficiency

1. **System Response Time**

- All user requests, such as submitting forms or querying the database, should be processed within **2 seconds 90% of the time**. For example, when a coordinator requests for pending tours, the results should be displayed almost instantly, ensuring a smooth and responsive experience.

2. **Page Load Speed**

- The platform must ensure that all pages load within **3 seconds** on a standard broadband connection of **5 Mbps**. For instance, when a guide logs in and navigates to their dashboard, the page showing their assigned tours should fully load in less than 3 seconds which avoids any noticeable delays.

3. **Data Processing**

- High school application forms must be processed by the system's prioritization algorithm within **5 seconds** of submission. For example, when a school submits a campus tour application, the system should *quickly sort and prioritize* the request based on predefined criteria, ensuring the application status is updated in real-time.

4. **Efficiency**

- The system should optimize its use of resources like memory and processing power to ensure smooth operation, especially during high traffic times. For instance, even when 50+ users (guides, coordinators, etc.) are actively using the system, there should be no significant performance drops or system slowdowns.

5. **Responsiveness and Performance**

- The system must provide quick feedback to user interactions, ensuring low latency during key operations. For example, when a candidate guide submits feedback for a completed tour, the system should immediately show a confirmation message, indicating successful submission without delays.

Reliability and Availability

1. **System Availability**

- The platform must maintain an uptime of **99.9%**, ensuring users can access it with minimal interruptions, even during peak usage times like the high school application season. For example, a coordinator should be able to manage tour schedules and guides should access their assignments without experiencing system downtime, especially during busy periods like the summer tour season.

2. **Backup and Recovery**

- Automated backups must occur every **12 hours**, with the system capable of recovering from a failure within **1 hour** (Recovery Time Objective - RTO). For example, if there's a system outage during tour scheduling, the platform should restore full functionality within an hour, ensuring no data or tours are lost in the process.

3. **Error Handling and Fault Tolerance**

- The system should gracefully handle errors by providing clear messages (e.g., "Network error, please try again in 2 minutes") and continue functioning in the event

of partial failures. For instance, if the real-time communication platform between guides experiences an issue, it should not affect other features like tour scheduling or feedback submission.

Scalability and Flexibility

1. **Horizontal Scalability**

- The system must scale horizontally to support up to **500 concurrent users** without degrading performance. For example, during peak times such as when multiple high schools are submitting applications or guides are logging in to view tours, the platform should continue to perform efficiently, even with hundreds of active users.

2. **Data Scalability**

- The system must handle an increasing volume of high school applications and user data without performance loss, initially supporting approximately **1,000 schools** and scaling to **5,000**. For example, as the number of schools applying for campus tours grows, the system should be able to efficiently manage the increasing data load, allowing coordinators and advisors to prioritize and approve applications quickly.

3. **Modular System Design**

- New features, such as additional reporting tools or new tour categories, should be added as independent modules without significant changes to existing ones. For example, if a new type of event like “virtual campus tours” is introduced, it should be implemented as a separate module without disrupting existing functionality.

Maintainability and Supportability

1. **Modular Architecture**

- The system should be built using a modular architecture which would allow each feature (e.g., Puantaj Page, High School Application Tracking System) to be updated independently. For example, if the Puantaj system requires an update or bug fix, this can be applied without affecting other critical functionalities like high school application tracking.

2. **Version Control**

- A popular version control system like **Git** must be used for all code changes, enabling rollbacks in case of critical issues. For instance, if a new update introduces a problem with tour scheduling, the development team should be able to revert to the previous stable version within minutes to minimize downtime. Git will also be beneficial for other types of branching and modular development of features.

3. **Code Documentation**

- All system modules should be thoroughly documented to make maintenance and team handovers easier. For example, if a new developer joins the project, they should be able to quickly understand how the tour approval algorithm works by referring to the updated and detailed documentation.

Ethical and Environmental Requirements

1. **Ethical Requirements**

- The system must handle all user data ethically, ensuring it is not shared or misused without consent. For example, when a guide submits feedback or personal details, the system should clearly inform them about how their data will be used, complying with GDPR and other relevant data privacy laws.

2. **Environmental Efficiency**

- The platform must be optimized to minimize its environmental impact, ensuring efficient use of computing resources. For instance, by optimizing the database and code to reduce unnecessary processing power, the system will consume fewer server resources, contributing to lower energy consumption and supporting sustainability goals.

Tech Stack

Frontend

React

React is a widely used front-end framework maintained by Facebook. It has strong community support and detailed documentation. Some of the key features of React:

- React uses a component-based architecture, which means developers can create reusable UI elements. It's like having a box of LEGO pieces – the same pieces can be used to build different structures, saving time and reducing errors.
- Efficient updates: React uses a Virtual DOM, which is like a blueprint of the webpage. When changes occur, React compares this blueprint to the actual webpage and only updates what's necessary. This makes the app faster and smoother for users.

- Easy to learn syntax: React uses JSX, which lets developers write HTML-like code directly in JavaScript. This makes the code more readable and intuitive, especially for those already familiar with HTML.
- One-way data flow: React manages data in a predictable way, flowing in one direction. This is like a river flowing downstream – it's easier to track where data is coming from and going to, making debugging simpler.

TypeScript

TypeScript adds static typing to JavaScript, catching errors early and improving code quality and maintainability.

- Improved teamwork: When working with others, TypeScript acts like a contract between developers. It clearly defines what kind of data should be used where and reduces misunderstandings.
- Catch errors early: TypeScript is like a spell-checker for code.

Redux

React state management can get complicated easily as the application grows over time. Redux is a solution that scales well for large applications.

- Central data storage: Redux provides a single place to store all the app's data, making organization easier.
- Great for complex apps: As the app grows more complex, Redux helps keep things organized.
- Predictable state changes: With Redux, data in the app changes in a very structured way. It's like having a clear, step-by-step recipe for updating the app's information, making it easier to understand and debug.

Backend

Node.js with Express.js

Express.js is a minimal and flexible web application framework built on top of Node.js. It comes with built-in solutions like middleware, multer, passport, and routing to ease development.

- Lightweight and flexible: Express.js is simple but powerful, allowing developers to add only the needed features without introducing too much complexity.
- Easy routing: Express makes it simple to direct web traffic to the right place in the application.
- Middleware support: Extra functionality can be easily added to the app using middleware. Think of it as a production line where steps can be added or removed as needed, customizing how the app handles requests and responses.

- Database-compatible: Express works well with any database of choice.
- Fast performance: Built on Node.js, Express is designed for speed. It can handle many users at once efficiently.

Database

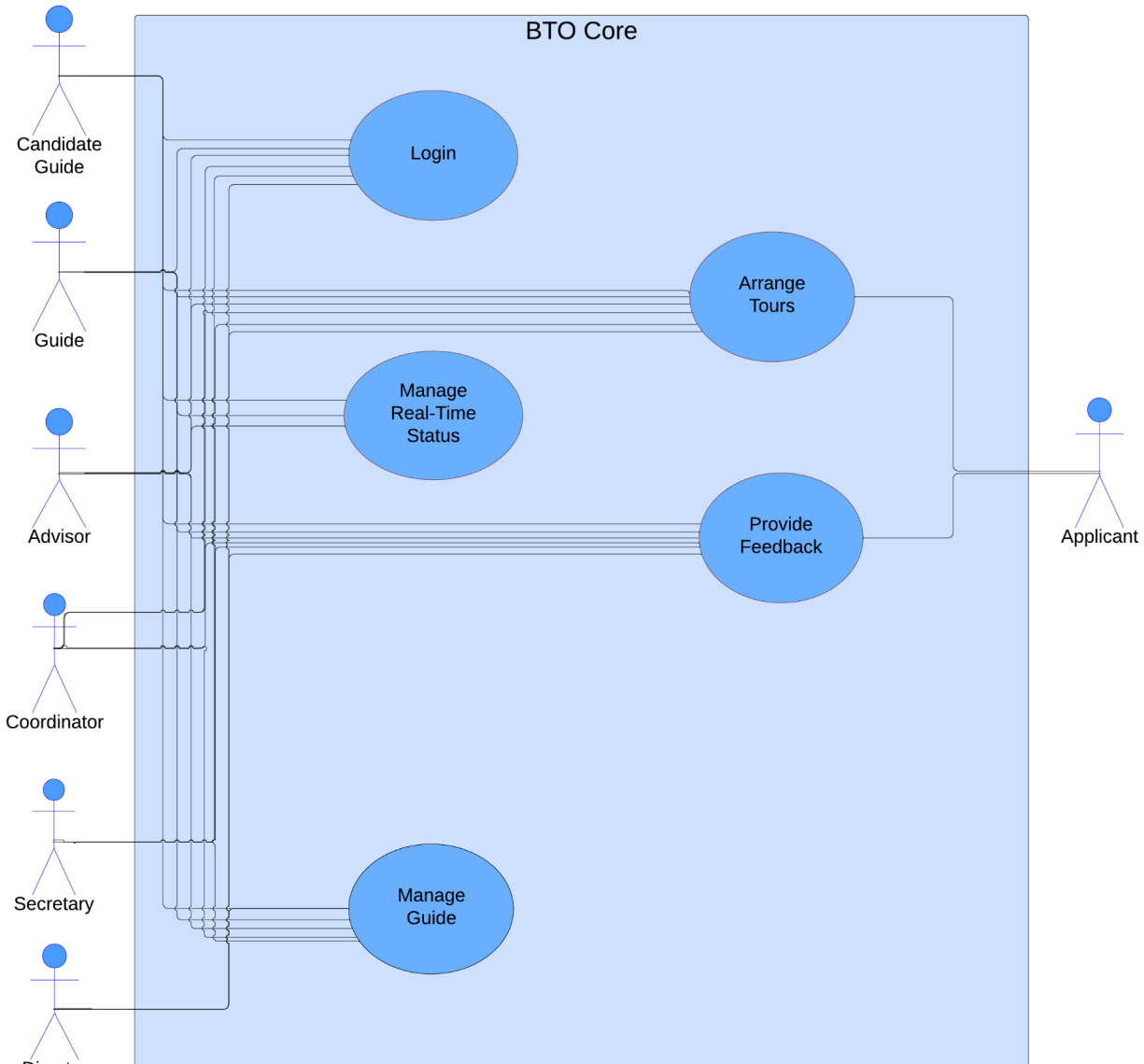
PostgreSQL

PostgreSQL is a widely used, powerful open-source relational database system. Some of its key features are:

- Reliable and stable: PostgreSQL is known for its reliability.
- Scalable: As an application grows, PostgreSQL can grow with it.
- Supports modern data types: PostgreSQL can work with newer types of data like JSON. This means it can adapt to changing data needs.

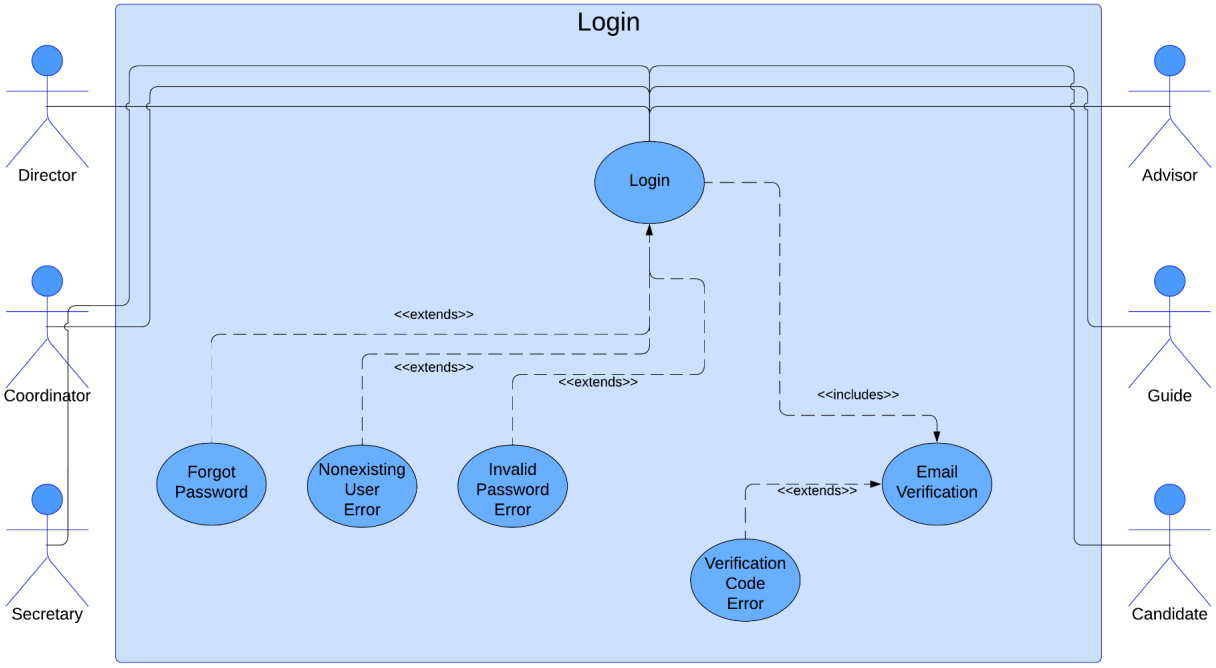
Use Case Diagrams

1. Level 0 Diagram

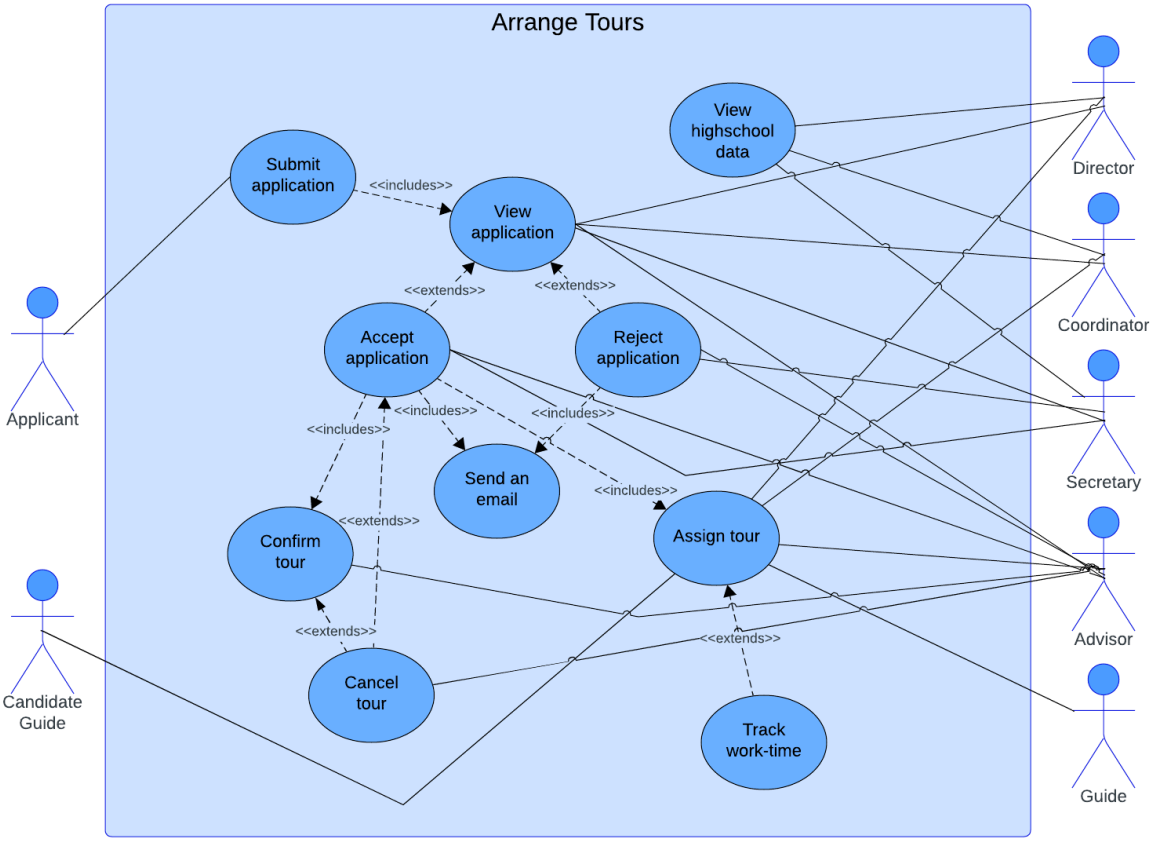


2. Level 1 Diagrams

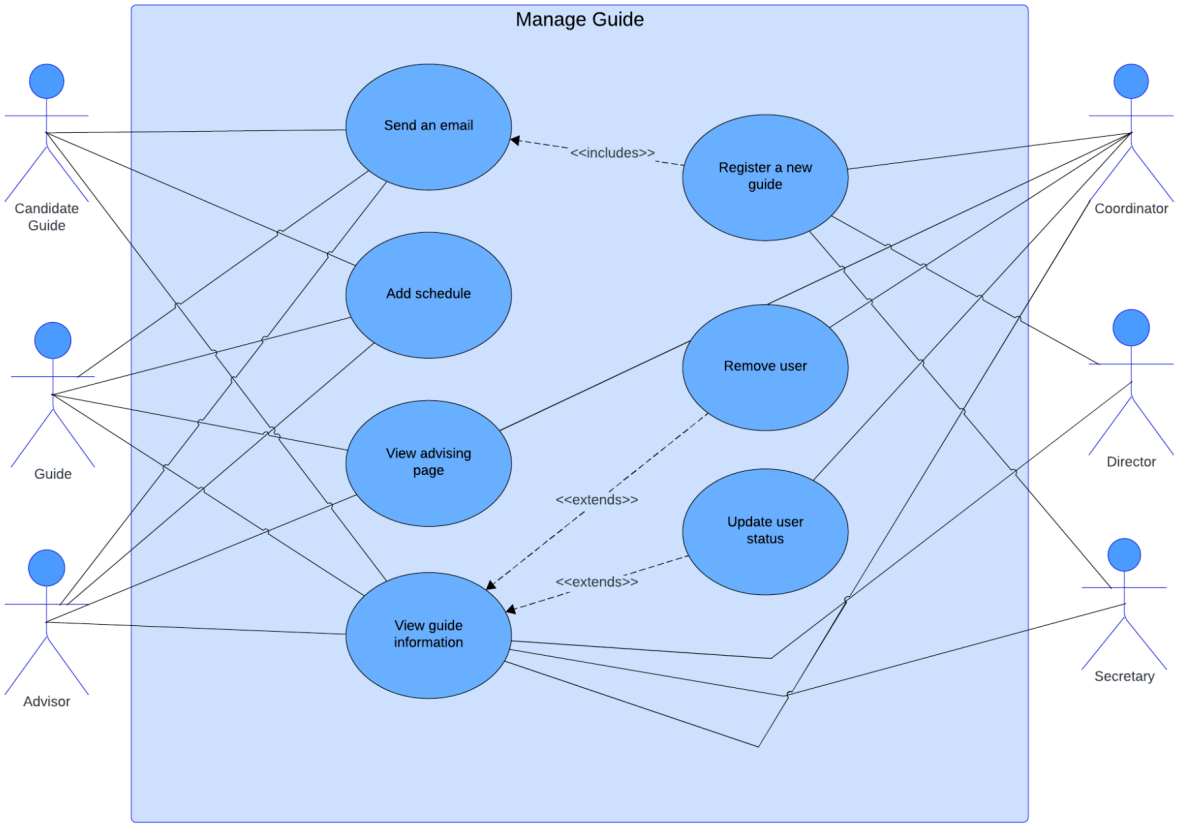
2.1. Login



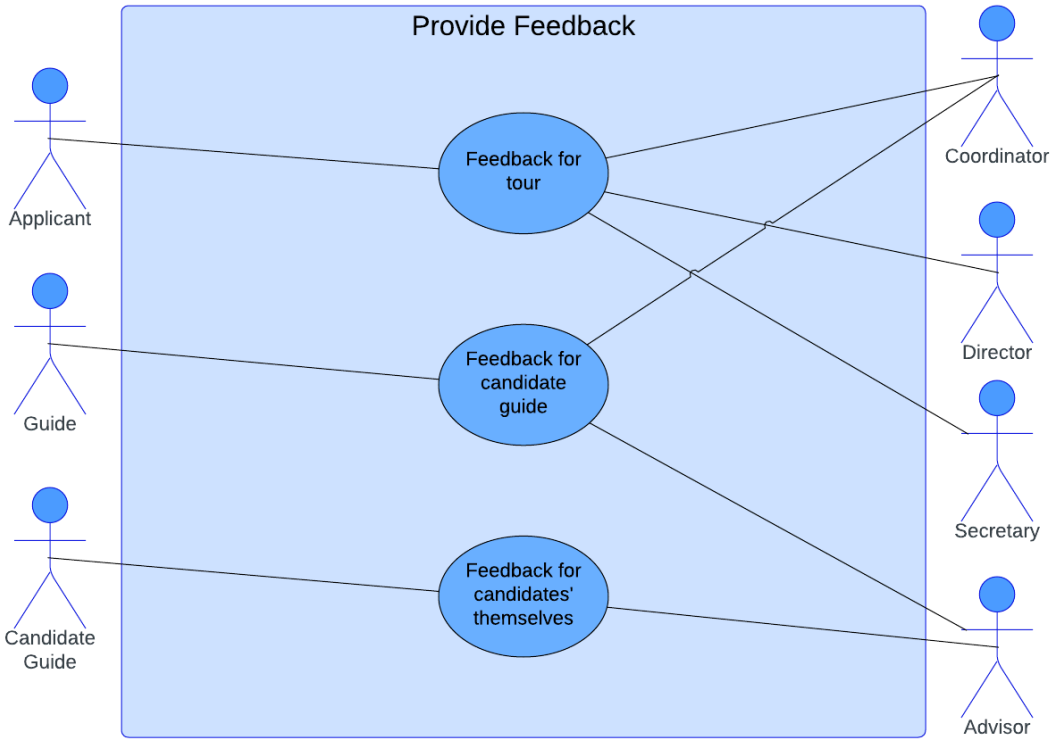
2.2. Arrange Tours



2.3. Manage Guide

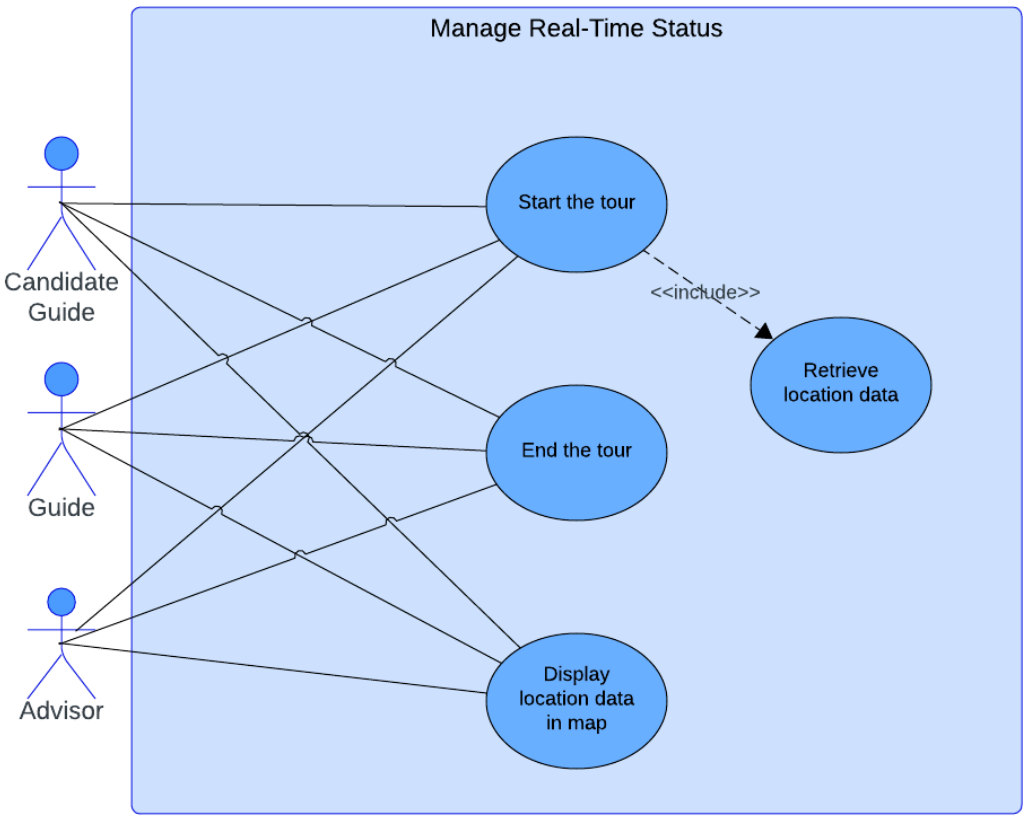


2.4. Provide Feedback



2.5. Manage Real-Time Status

.



USER STORIES

User Story 1: {1}

As a director, I want to log in to the system so that I can start my work session.

Acceptance Criteria:

- Users should be able to log in with valid credentials, which are their email address and password. Invalid login attempts should return appropriate error messages.
- These messages indicate if the user entered the wrong password or if the user's email address is incorrect, meaning it does not exist in the system.

User Story 2: {1}

As a coordinator, I want to reset my password when I forget it so that I can regain access to the system.

Acceptance Criteria:

- Users should be able to initiate a password reset via email.

- A reset link should be sent to the email to be entered.
- The system should require new passwords to meet security criteria.

User Story 3: {1}

As a candidate guide, I want to log in to the system so that I can start my work session.

Acceptance Criteria:

- Users should be able to log in with valid credentials, which are their email address and password. Invalid login attempts should return appropriate error messages.
- These messages indicate if the user entered the wrong password or if the user's email address is incorrect, meaning it does not exist in the system.

User Story 4: {1}

As a guide, I want to reset my password when I forget it so that I can regain access to the system.

Acceptance Criteria:

- Users should be able to initiate a password reset via email.
- A reset link should be sent to the email to be entered.
- The system should require new passwords to meet security criteria.

User Story 5: {1}

As a secretary, I want to log in to the system so that I can start my work session.

Acceptance Criteria:

- Users should be able to log in with valid credentials, which are their email address and password. Invalid login attempts should return appropriate error messages.
- These messages indicate if the user entered the wrong password or if the user's email address is incorrect, meaning it does not exist in the system.

User Story 6: {1}

As an advisor, I want to reset my password when I forget it so that I can regain access to the system.

Acceptance Criteria:

- Users should be able to initiate a password reset via email.
- A reset link should be sent to the email to be entered.
- The system should require new passwords to meet security criteria.

User Story 7: {2}
As an applicant, I want to submit an application for a tour to begin the process of joining the campus tour.

Acceptance Criteria:

- Applicants should be able to access an application form.
- The form should capture the details about the school (e.g., school name, number of participants, preferred dates, email of the applicant,etc.)
- Applicants should receive a confirmation message after submitting the form.

User Story 8: {2}

As an advisor, I want to view submitted applications to approve or reject them.

Acceptance Criteria:

- Advisor should be able to view a list of submitted applications.
- Advisors should have the option to approve, reject, or mark applications as pending with reasons.

User Story 9: {2}

As an advisor, I want to approve or reject applications so that I can manage tour scheduling.

Acceptance Criteria:

- Approved applications must automatically notify the secretary for confirmation.
- After the confirmation of the secretary, automated email should be generated to inform the applicant.
- Rejected applications should generate an automated email to inform the applicant.

User Story 10: {2}
As a guide, I want to assign myself to tours according to my availability.

Acceptance Criteria:

- The table should clearly highlight unassigned tours, making it easy to identify tours that require immediate attention.
- Guides should have access to a dedicated dashboard displaying all unassigned tours, along with key details such as; date and time of the tour, school name, number of participants, location of the tour.
- Guides can filter tours by date, time, or number of participants to match their availability and preference.
- Upon successful assignment, the system should associate the tour with the guide.

- Guides should be able to view their assigned tours in a separate section of the dashboard for efficient schedule tracking.

User Story 11: {2}

As an advisor, I want to update the status of a tour if the applicant cancels or the tour is canceled for any reason so that the system reflects accurate and up-to-date information.

Acceptance Criteria:

- Advisors should have access to a Tour page and can update the current status of the tour as “cancelled”.
- Tour Page and Review Tour Page are updated to reflect the new status.

User Story 12: {2}
As an advisor, I want to assign my candidate guide to a specific tour so that they can complete their required training.

Acceptance Criteria:

- Advisor can see the quota for candidate guides and assign them.

User Story 13: {2}
As a guide, I want to record my tour details and the hours I worked on the score sheet so that my contributions are documented for payroll and performance tracking.

Acceptance Criteria:

- Guides should have access to a Puantaj Page from their dashboard.
- Guides must be able to input the following details for each tour: school name, date, time, working hours.
- Once the guide completes their entries, they should have a “Submit” button to log their work hours.

User Story 14: {2}
As a director, coordinator, or secretary, I want to view statistical insights such as tour distribution by day, total number of tours, cancellation statistics, and tour distribution by city so that I can monitor performance and make data-driven decisions.

Acceptance Criteria:

- Users must have access to a dedicated Data page from their dashboard.
- The dashboard must display the following key metrics: tour distribution by day, the number of tours, cancellation statistics: tour distribution by city.
- Users should be able to filter the data based on the following criteria: date range (e.g., week, month, year), specific cities or regions, etc.
- Users must be able to export the data in formats like Excel, PDF, or CSV for further analysis.

User Story 15: {2}
As a director, coordinator, or secretary, I want to view the tour tables so that I can oversee the current status of tours and ensure proper assignment and scheduling.

Acceptance Criteria:

- Users must have access to the Tour Table from their dashboard.
- The table should display the following details for each tour: Tour ID, School name, City, Date and time, Number of participants, Assigned guide(s) (if any), Status (e.g., Unassigned, Assigned, Completed, Cancelled).
- Users should be able to filter the table.
- A search bar should allow users to quickly find tours by school name, or guide name.

User Story 16: {2}
As a director, coordinator, or secretary, I want to view the review tour tables so that I can oversee the changes made by advisors and ensure the accuracy of the tour details.

Acceptance Criteria:

- Users must have access to a Review Tour Table page from their dashboard.
- This page should display all tours currently in the review process or modified by advisors.

User Story 17: {3}
As a coordinator, I want to register a new guide so that new candidate guides can start using the application.

Acceptance Criteria:
- The coordinator should be able to fill out a form with required fields.
- The system should validate the input fields and show error messages for missing or invalid data.
- Once the form is submitted, the new guide's information should be saved in the database.

User Story 18: {3}
As a candidate guide, I want to get an email about my login information so that I can start using the application.

Acceptance Criteria:
- The system should send an email to the candidate guide's registered email address upon successful registration.

User Story 19: {3}
As a candidate guide, I want to add my schedule to the application so that when there is a tour to be guided in my spare time, my advisor could inform me about it.

Acceptance Criteria:
- Candidate guides should be able to access the scheduling page after logging in.

- The scheduling page should provide a weekly calendar interface for selecting available time blocks.
- The system should allow the user to save, edit, and delete their schedule.

User Story 20: {3}
As an advisor, I want to view guide information so that when a tour is not taken, I can view guides' schedules and phone numbers to contact them to ask if they can guide the tour.

Acceptance Criteria:
- Advisors should have access to a dashboard listing all candidate guides they oversee.
- The dashboard should display each guide's name, phone number, and schedule.
- Advisors should be able to filter guides by availability.
- The information displayed should update dynamically based on changes made by the guides.

User Story 21: {3}
As a coordinator, I want to remove users so that they cannot access the application when they graduate or leave BTO.

Acceptance Criteria:
- The coordinator should have access to a user management page listing all users.
- Each user entry should include an option to remove the user.
- When a user is removed, their status should be updated to "Inactive" in the database.

User Story 22: {3}
As a coordinator I want to update users' status so that they can access different features when they are promoted or demoted.

Acceptance Criteria:
- Coordinators should be able to select a user and update their status via a dropdown or selection field.
- Status options should include predefined roles (e.g., Candidate Guide, Guide, Advisor, Coordinator).
- The system should validate changes and reflect the updated status immediately in the database.
- Role-specific permissions should update dynamically based on the new status.

User Story 23: {3}
As a coordinator, I want to view the advising page so that I can see which advisor is responsible for which candidate guide.

Acceptance Criteria:
- The advising page should display a list of all advisors and their assigned candidate guides.
- Each advisor's section should show a table of candidate guides, including their names, bilkent_ids, and statuses.
- Coordinators should be able to filter the list by advisor or guide name.

- The page should update dynamically to reflect any reassignment of guides to advisors.

User Story 24: {4}
As an applicant, I want to give feedback about the organized tour to express my evaluations.

Acceptance Criteria:

- The system must allow applicants to access a feedback form after their tour.
- A link which will direct the users to the feedback page should be sent to the applicants.
- The feedback should be submitted as text or pdf.
- Confirmation of feedback submission must be displayed to the applicant.
- Feedback should be stored in the system and made available to relevant roles.

User Story 25: {4}
As a candidate guide, I want to give feedback about my experience about the tour so that my advisor will assess my perspective.

Acceptance Criteria:

- The system should allow candidate guides to give feedback after completing their tour.
- The feedback should be submitted as text or pdf.
- A confirmation message should be displayed upon successful submission of feedback.
- Feedback must be associated with the specific tour.
- Feedback submitted by candidate guides should only be accessible by the advisor and relevant roles.

User Story 26: {4}
As a guide, I want to provide feedback for how well the candidate guide did on his/her assessed tour so that other guides including the advisor and the coordinator will track the progress of the candidate.

Acceptance Criteria:

- The system must provide a feedback interface for guides to evaluate candidate guides.
- The feedback should be submitted as text or pdf.
- Submitted feedback must be associated with the specific candidate guide and tour.
- The feedback should be stored and viewable by authorized roles, preserving confidentiality.

User Story 27: {4}

As a coordinator, I want to view the feedback given for a candidate guide to monitor the performance of the candidate guide.

Acceptance Criteria:

- The system must provide a dashboard or section for viewing feedback related to candidate guides.
- Feedback must be filterable by candidate guide, date, or tour.
- Only feedback submitted by guides, applicants, and advisors should be accessible in this view.
- The system must ensure the confidentiality of the feedback.

User Story 28: {4}
As a coordinator, I want to view the applicant feedback given for a tour so that I will assess the approach BTO takes during tours.

Acceptance Criteria:

- The system must allow the coordinator to access all applicant feedback related to tours.
- Feedback must be filterable by date, location, or tour type.
- Feedback visibility must comply with data privacy policies.

User Story 29: {4}

As a director, I want to view the applicant feedback given for a tour so that I will make decisions about the further tours.

Acceptance Criteria:

- The system must allow the director to access aggregated applicant feedback from multiple tours.

User Story 30: {4}
As a secretary, I want to view the applicant feedback given for a tour so that I will assess the success of the tour.

Acceptance Criteria:

- The system must allow the secretary to view applicant feedback associated with specific tours.

User Story 31: {4}

As an advisor, I want to view the self-feedback of the candidate guide so that I will evaluate his/her performance

Acceptance Criteria:.

- The system must allow the advisor to view self-feedback submitted by candidate guides.
- Feedback must be associated with the specific guide and tour.
- Self-feedback should include timestamps and tour context.

User Story 32: {4}

As an advisor, I want to enter my observations about the candidate guide as a feedback to the system so that we will collect performance data of the candidate.

Acceptance Criteria:.

- The system must provide a section for advisors to submit observations about candidate guides.
-
- Submitted feedback must be linked to the candidate guide and specific tour.
- Feedback must be stored and made accessible to relevant roles (e.g., Coordinator, Director).
- Advisors must receive confirmation upon successful submission of their observations.

User Story 33: {5}

As a Guide/Candidate Guide, I want to start a tour and update its status in real-time So that others can track the tour group's progress and location

Acceptance Criteria:

- I should be able to select a specific tour point (Library, Faculty Buildings, etc.)
- I should be able to mark a point's status as "Upcoming," "Ongoing," or "Done"
- The system should automatically retrieve and update my current location
- Each tour point should have a unique ID displayed
- The status updates should be reflected immediately on the map view

User Story 34: {5}

As an Advisor, I want to view the real-time status and location of ongoing tours So that I can monitor tour progress and coordinate effectively

Acceptance Criteria:

- I should see all active tours on the map view
- I should see the current status of each tour point (Done, Ongoing, Upcoming)
- The map should show the real-time location of tour groups
- I should be able to distinguish between different tour groups on the map
- The information should update automatically without manual refresh

User Story 35: {5}

As a Guide/Candidate Guide, I want to end a tour session So that the system stops tracking and displaying my location

Acceptance Criteria:

- I should be able to officially end a tour when completed
- The system should stop tracking my location once the tour ends
- All tour points should be marked as "Done" upon tour completion
- The tour status should be archived for future reference
- Other users should no longer see my tour group on their map view