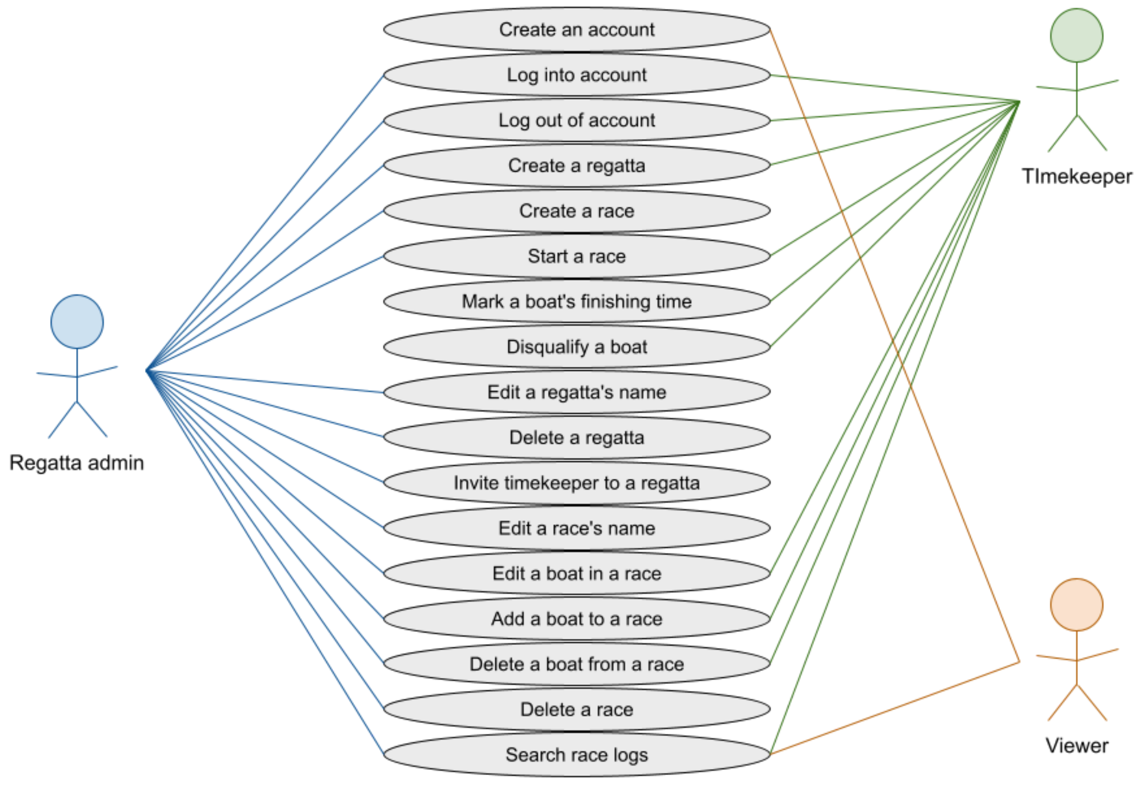
# Assignment 2: Use Cases

**Project:** flotE

**Group 1:** Dila Ozersen, Macy Graves, Ashley Bhandari, Austin Henlotter, Jamie Denley, Anushka Trehan (Manager)

**Project description:** The current method for tracking finishing times in a regatta is fairly archaic: an admin hand writes the order they spot racers crossing the finish line. The finish line for a regatta is not overly visible so viewers have to wait for the standings to be posted. We are digitalising this process so that viewers have a more current knowledge of finishing orders. Our application will allow some admin to create a race and mark when racers finish. Viewers will be able to see this log in real time and be able to access stored race data from previous events. Logs of previous races will be stored and organized by regatta.

## Use Case Diagram



## Use Cases

**Tentative database schema:**

- User table
  - Primary key: User ID
  - Fields: Email, Password, Regatta ID list (as admin)
- Regatta table
  - Primary key: Regatta ID
  - Fields: Regatta name
- Race table
  - Primary key: Race ID
  - Foreign keys: Regatta ID
  - Fields: Name, Start time, { Boat ID, Finish time } list
- Participant table
  - Primary key: Participant ID
  - Fields: Name, { Boat ID, Race ID } list

**TODO:** Slideshow to present use cases

| Create a new account | |
|---|---|
| Description | A new user should be able to create an account so that they can create regattas/races and timekeep. |
| Actors | Primary: Unregistered user |
| | Secondary: DB |
| Goal | Create a new account and log the user in |
| Preconditions | User logged out and on the landing page |
| Postconditions | Success: User creates an account |
| | Failure: The account cannot be created |
| Exceptions | Invalid email, invalid password, existing account, internal error |
| Trigger | User clicks the "Create an account" button |
| Main success scenario | 1. User enters their email and password<br>2. Application saves credentials to the DB<br>3. Application confirms successful account creation<br>4. User is logged in and directed into the application |
| Error scenarios | 1a. Email is invalid. An error message indicating this is displayed.<br>1b. Password is invalid (weak, contains spaces, etc.). An error message describing password requirements is displayed.<br>2a. A user with the inputted email already exists in the DB. An error message indicating this is displayed.<br>2b. Account cannot be created due to an internal error. A "something went wrong" error is displayed. |

| **Log an existing user in** | |
|---|---|
| Description | Log an existing user into their account |
| Actors | Primary: Existing user |
| | Secondary: DB |
| Goal | Log the user in |
| Preconditions | User is logged out and on the landing page |
| Postconditions | Success: User is logged in and directed to the application |
| | Failure: Login attempt fails |
| Exceptions | Email does not exist in the DB, password is incorrect, internal error |
| Trigger | User clicks “Login” button |
| Main success scenario | 1. User enters their email and password<br>2. Application verifies credentials and saves an auth token to local storage<br>3. Application confirms successful login<br>4. User is directed into the application |
| Error scenarios | 2a. Email does not exist in the DB. An "account with that email or password does not exist" error message is displayed.<br>2b. Password is incorrect. An "account with that email or password does not exist" error message is displayed.<br>2c. User cannot be logged in due to an internal error. A "something went wrong" error message is displayed. |

| **Log a user out of their account** | |
|---|---|
| Description | Log a user out of their account |
| Actors | Primary: Existing user |
| Goal | Log the user out |
| Preconditions | User is logged in |
| Postconditions | Success: User is logged out and directed to the landing page |
| | Failure: User cannot be logged out |
| Exceptions | Logout fails, redirect fails |
| Trigger | User clicks “Log out” button |
| Main success scenario | 1. Auth token is removed from local storage<br>2. User is directed to the landing page |
| Error scenarios | 1a. Logout fails. Error message indicating this is displayed.<br>2a. Redirect fails. User stays on same page |
| Alternative scenarios | 1. Auth token expires and the user is automatically logged out |

| Create a regatta | |
|---|---|
| Description | Create a new regatta with the user as its admin, to which races can later be added |
| Actors | Primary: User |
| | Secondary: DB |
| Goal | Create a new regatta and add it to the user's "regatta list" |
| Preconditions | User is logged in and on the home page |
| Postconditions | Success: A regatta is created with the user as its admin |
| | Failure: Application fails to create the regatta |
| Exceptions | Email does not have an associated account, DB error |
| Trigger | User presses the “Create a regatta” button |
| Main success scenario | 1. User inputs the regatta name<br>2. User invites timekeepers by email<br>3. User clicks the "Save" button<br>4. Application saves regatta and assigns it to user in the DB<br>5. Application confirms successful regatta creation and creates a page for the regatta<br>6. User is directed to the regatta's page |
| Error scenarios | 2a. The inputted email does not have an associated account. An error message indicating this is displayed.<br>4a. The regatta cannot be saved due to an internal error. A "something went wrong" error message is displayed. |

| Create a race | |
|---|---|
| Description | Create a new race, assign it to a regatta, and add a list of registered participants |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | To add a race that participants can add boats to and compete in, to the overall regatta. |
| Preconditions | User is logged in and on a regatta's page |
| Postconditions | Success: Race is created and added to the regatta |
| | Failure: Application fails to create the race |
| Exceptions | Repeated IDs in list, DB error |
| Trigger | User clicks the "Add a race" button |
| Main success scenario | 1. User inputs the race name<br>2. For each boat in the race, user enters the associated ID and participant names<br>3. User clicks the "Save" button<br>4. Application saves the race and adds it to each participant's record in the DB<br>5. Application confirms successful save and creates a page for the race<br>6. User is directed to the race's page |
| Error scenarios | 1a. A boat with the inputted ID has already been added to the list. An error message indicating this is displayed.<br>4a. Race cannot be saved to to an internal error. A "something went wrong" error message is displayed. |

| **Start a race** | |
|---|---|
| Description | A timekeeper starts the race, beginning the stopwatch |
| Actors | Primary: Timekeeper or regatta admin (acting as a timekeeper) |
| | Secondary: DB |
| Goal | Start a race and record its start time |
| Preconditions | User is logged in and on a race's page. |
| Postconditions | Success: A list of boat ID's and their finishing times is logged |
| | Failure: Boat ID's and their finishing times are not logged |
| Exceptions | No registered boat, race already started, stopwatch fails, DB error |
| Trigger | The user clicks the "Start race" button |
| Main success scenario | 1. User is directed to the race's timekeeping page<br>2. A stopwatch begins<br>3. The race's start time is saved to the DB |
| Error scenarios | 1a. No boats are registered for the race. An error message indicating this is displayed.<br>1b. Another user has already started the race. An error message indicating this is displayed.<br>2a. The stopwatch does not begin. A "something went wrong" error message is displayed.<br>3a. The start time is not saved to the DB. A "something went wrong" error message is displayed. |

| Mark when a boat finishes the race | |
|---|---|
| Description | A timekeepers marks when a boat finishes the race, saving the timestamp and boat's ID |
| Actors | Primary: Timekeeper |
| | Secondary: DB, SMS API |
| Goal | Save the current timestamp and boat's ID |
| Preconditions | User is logged in, has started a race, and is on the timekeeping page |
| Postconditions | Success: The boat's finishing time and ID are saved |
| | Failure: The boat's finishing time and ID are not saved |
| Exceptions | SMS request fails |
| Trigger | User clicks the "Mark" button |
| Main success scenario | 1. A row is added to the "Finishing times" list with the current timestamp and an empty "ID" field<br>2. User fills out the ID field; selectable suggestions from the race's list of registered boats appear as they type<br>3. User clicks the "Save" button<br>4. The ID and timestamp are sent to the server via SMS and added to the race's page |
| Error scenarios | 4a. Info cannot be sent due to an internal error. Application periodically reattempts request until it is sent. |
| Alternative scenarios | 2. The inputted ID does not exist in the list. User manually finishes typing it in.<br>3. If all IDs have been accounted for, the stopwatch stops running and a "race completed" message appears to the user and on the race's page |

| **Disqualify a boat from a race** | |
|---|---|
| Description | A boat should be disqualified if it did not start or finish the race |
| Actors | Primary: Timekeeper |
| | Secondary: DB, SMS API |
| Goal | Disqualify the boat from the race and indicate whether it is a DNS or DNF |
| Preconditions | User is logged in, has started a race, and is on the timekeeping page |
| Postconditions | Success: The boat is disqualified from the race |
| | Failure: The boat remains in the running |
| Exceptions | Specified boat ID is not in the race, SMS request fails |
| Trigger | User clicks the "Add DNS/DNF" button |
| Main success scenario | 1. User inputs the boat ID and selects whether it was a DNS or DNF<br>2. User clicks the "Save" button<br>3. The ID and result are sent to the server via SMS and added to the race's page |
| Error scenarios | 1a. Specified ID is not registered for the race. An error message indicating this is displayed.<br>3a. Info cannot be sent due to an internal error. Application periodically reattempts request until it is sent. |

| **Edit a regatta** | |
|---|---|
| Description | A regatta's admin can edit the regatta's name and date |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | Change the regatta’s name |
| Preconditions | User is logged in, has created a regatta, and is on that regatta's page |
| Postconditions | Success: The regatta is successfully edited |
| | Failure: Application fails to edit the regatta information |
| Exceptions | DB error |
| Trigger | User clicks the “Edit” button |
| Main success scenario | 1. User inputs the new name into the provided field<br>2. User presses "Save" button<br>3. Application saves the new name to the DB<br>4. Application confirms successful name change |
| Error scenarios | 3a. The name is not saved to the DB. A "something went wrong" error message is displayed. |
| Alternative scenarios | 2. User clicks the "Cancel" button. Edits are not saved. |

| **Delete a regatta** | |
|---|---|
| Description | A regatta's admin can delete the regatta (along with all races in the regatta) |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | That regatta's admin can delete their existing regatta. |
| Preconditions | User is logged in, has created a regatta, and is on that regatta's page |
| Postconditions | Success: The regatta is successfully deleted from the DB |
| | Failure: The regatta is not deleted from the DB |
| Exceptions | DB error |
| Trigger | The user clicks the "Delete regatta" button |
| Main success scenario | 1. A popup confirming that the user wants to delete the regatta and all its races is displayed.<br>2. User clicks the "Confirm" button<br>3. Application removes the regatta from the DB.<br>4. Application confirms successful deletion of the regatta |
| Error scenarios | 3a. The regatta is not removed from the DB properly. A "something went wrong" error message is displayed. |
| Alternative scenarios | 2. The user does not want to delete the regatta and clicks the "Cancel" button. The regatta is not deleted. |

| Invite a timekeeper to a regatta | |
|---|---|
| Description | A regatta's admin can invite other users by email to act as a timekeeper for races in the regatta |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | Invite another user to timekeep for a regatta by sending an email. |
| Preconditions | User is logged in, has created a regatta, and is on that regatta's page |
| Postconditions | Success: Timekeepers are given access to timekeeping features for the regatta's races |
| | Failure: Timekeepers cannot access timekeeping features for the regatta's races |
| Exceptions | User associated with email does not exist, timekeepers don't gain access, users aren't notified of changes |
| Trigger | User clicks the "Invite timekeepers" button |
| Main success scenario | 1. User inputs one or more emails and invites the people associated with the emails.<br>2. Application grants users associated with those emails access to the regatta's timekeeping features<br>3. Application notifies user and invited users of success |
| Error scenarios | 1a. User associated with an inputted email does not exist. An error message indicating this is displayed.<br>2a. Application fails to grant the invitees access. An error message indicating this is displayed.<br>3a. Application fails to notify user and invitees of success. |
| Alternative scenarios | 1. User clicks the "Cancel" button. Edits are not saved. |

| Edit a race's name | |
|---|---|
| Description | A regatta's admin can change the race's name |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | Rename a race |
| Preconditions | User is logged in, has created a regatta, and is the page of a race that belongs to the regatta |
| Postconditions | Success: The race is successfully renamed |
| | Failure: Application fails to rename the race |
| Exceptions | Invalid input, DB error |
| Trigger | User clicks the "Edit" button |
| Main success scenario | 1. User types a new name in<br>2. User clicks the "Save" button<br>3. Application saves new name to the DB<br>4. Application confirm successful name change |
| Error scenarios | 1a. Invalid input. An error message indicating name requirements is displayed.<br>3a. Name is not saved to the DB. A "something went wrong" error is displayed. |
| Alternative scenarios | 2. User clicks the "Cancel" button. Edits are not saved. |

| Edit a boat in a race | |
|---|---|
| Description | A regatta's admin or the race's timekeeper can edit a registered boat's ID, its associated participants' names, and the boat's finishing time (if the race has been completed) |
| Actors | Primary: Regatta admin, timekeeper |
| | Secondary: DB |
| Goal | User edits a registered boat's ID, participants' names, or finishing time |
| Preconditions | User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page |
| Postconditions | Success: User successfully edits a registered boat's ID, participants' names, or finishing time |
| | Failure: User fails to edit a registered boat's ID, participants' names, or finishing time |
| Exceptions | DB error |
| Trigger | The user Clicks the “Edit” button located next to that specific boat they wish to edit |
| Main success scenario | 1. The user edits the attribute that they wish to change (the ID, the name, and/or the finishing time)<br>2. The user clicks the "Save changes" button to save the information<br>3. Application updates the DB<br>4. Application notifies the user of success |
| Error scenarios | 3a. The application fails to update the information properly. A "something went wrong" error message is displayed.<br>4a. The application fails to let the user know it succeeded. |
| Alternative scenarios | 2. User clicks the "Cancel" button. Edits are not saved. |

| Add a boat to a race | |
|---|---|
| Description | A regatta's admin or the race's timekeeper can register a new boat to the race |
| Actors | Primary: Regatta admin, timekeeper |
| | Secondary: DB |
| Goal | The user adds a new boat to the race |
| Preconditions | User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page |
| Postconditions | Success: The user adds a boat to the race |
| | Failure: The user is unable to add boats to the race |
| Exceptions | Boat is already registered, DB error |
| Trigger | User clicks the “Add boat” button |
| Main success scenario | 1. User inputs the boat's name and its participants' names<br>2. Application saves the new boat to the DB<br>3. Application confirms success and the new boat is added to the list of boats in that particular race |
| Error scenarios | 1a. The user’s boat ID already exists in the race. An error message indicating this is displayed.<br>2a. The new boat is not saved. A "something went wrong" error is displayed. |

| Delete a boat from a race | |
|---|---|
| Description | A regatta's admin or the race's timekeeper can delete a registered boat from the race |
| Actors | Primary: Regatta admin, timekeeper |
| | Secondary: DB |
| Goal | User removes a registered boat from the race |
| Preconditions | User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page |
| Postconditions | Success: The user successfully deletes a boat from the race |
| | Failure: The user is unable to delete a boat from the race |
| Exceptions | DB error |
| Trigger | User clicks the "Delete boat" button |
| Main success scenario | 1. A popup confirming that the user wants to delete the boat is displayed<br>2. User clicks the "Confirm" button<br>3. Application removes the boat from the DB.<br>4. Application confirms successful deletion of the boat |
| Error scenarios | 3a. The regatta is not removed from the DB properly. A “something went wrong” error message is displayed. |
| Alternative scenarios | 2. The user does not want to delete the boat and clicks the "Cancel" button. The boat is not deleted. |

| Delete a race | |
|---|---|
| Description | A regatta's admin can delete a race, removing it from the DB and regatta |
| Actors | Primary: Regatta admin |
| | Secondary: DB |
| Goal | User deletes the race from both the DB and the regatta. |
| Preconditions | User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page |
| Postconditions | Success: User successfully deletes the race |
| | Failure: User fails to delete the race. |
| Exceptions | DB error |
| Trigger | The user pressed the “Delete race” button |
| Main success scenario | 1. A popup confirming that the user wants to delete the race is displayed<br>2. User clicks the "Confirm" button<br>3. Application removes the race from the DB.<br>4. Application confirms successful deletion of the race |
| Error scenarios | 3a. Race is not removed from the DB. A "something went wrong" error message is displayed. |
| Alternative scenarios | 1. The race is in progress. The message confirms that the user wants to delete the race while it's in progress.<br>3. The race is in progress. Application ends the race and timekeepers are kicked out. |

| Search for regattas and races | |
|---|---|
| Description | Search logs by regatta or race name. Clicking a result leads to that regatta or race's page. |
| Actors | Primary: All stakeholders (regatta admins, timekeepers, and viewers) |
| | Secondary: DB |
| Goal | Search logs for current and past regattas and races |
| Preconditions | User is on the landing page |
| Postconditions | Success: List of matching results is displayed |
| | Failure: The search fails to execute |
| Exceptions | DB error |
| Trigger | User clicks the search bar |
| Main success scenario | 1. User inputs their search query (regatta name / race name / participant name / boat ID) and clicks "Search" or enter button<br>2. Application queries DB to find matching results<br>3. A list of results is displayed |
| Error scenarios | 2a. DB query fails due to an internal error. A "something went wrong" error message is displayed. |
| Alternative scenarios | 3. No matching results. A message indicating this is displayed. |