flotE

timekeeping made easy

Group 1

Dila Ozersen

Macy Graves

Ashley Bhandari

Austin Henlotter

Jamie Denley

Anushka Trehan (Manager)

About the project

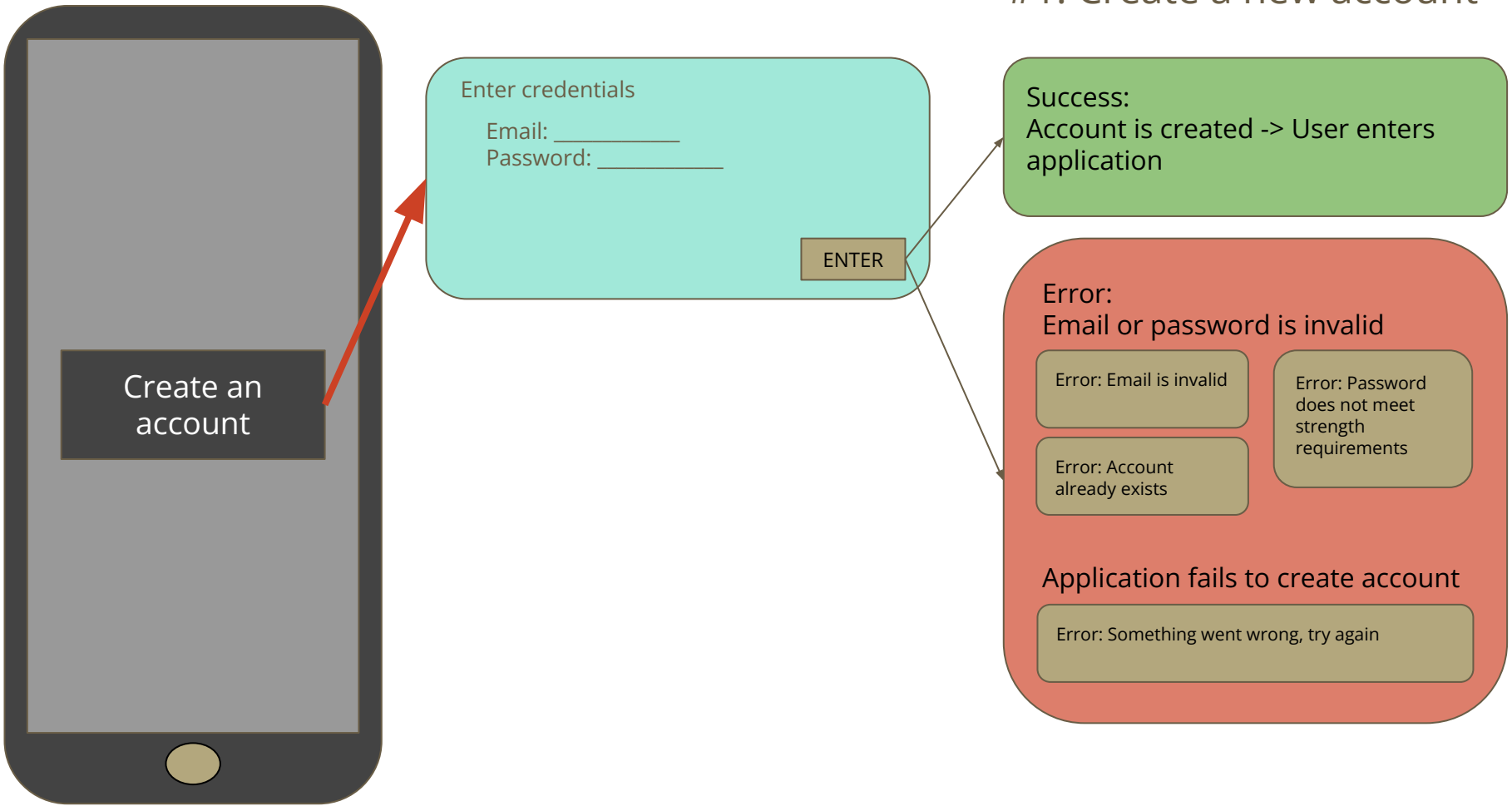
The current method for tracking finishing times in a regatta is fairly archaic: an admin hand writes the order they spot racers crossing the finish line. The finish line for a regatta is not overly visible so viewers have to wait for the standings to be posted. We are digitalising this process so that 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 cases


Create an account
Log into account
Log out of account
Create a regatta
Create a race
Start a race
Mark a boat's finishing time
Disqualify a boat
Edit a regatta's name
Delete a regatta
Invite timekeeper to a regatta
Edit a race's name
Edit a boat in a race
Add a boat to a race
Delete a boat from a race
Delete a race
Search race logs
Regatta admin
TImekeeper
Viewer

Use case #1: Create a new account

#1: Create a new account



Use case #1: Create a new account

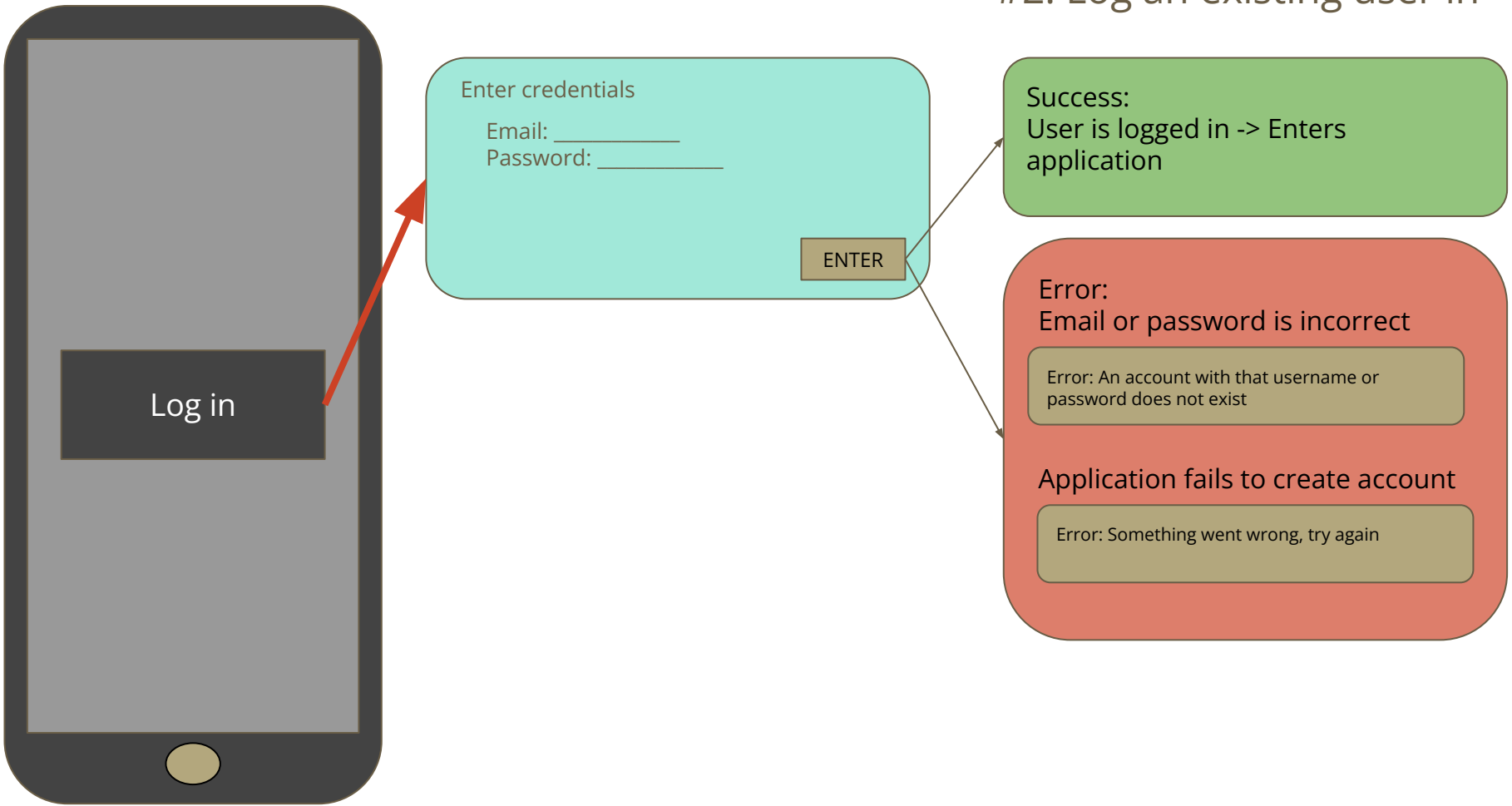
Description	A new user should be able to create an account so that they can create regattas/races and timekeep.
Actors	Primary: Unregistered user; Secondary: DB
Goal	Create a new account and log the user in
Preconditions	User logged out and on the landing page
Postconditions	Success: User creates an account; Failure: The account cannot be created
Exceptions	Invalid email, invalid password, existing account, internal error
Trigger	User clicks the "Create an account" button

Use case #1: Create a new account

Main success scenario	Error scenarios
1. User enters their email and password 2. Application saves credentials to the DB 3. Application confirms successful account creation 4. User is logged in and directed into the application	1a. Email is invalid. An error message indicating this is displayed. 1b. Password is invalid (weak, contains spaces, etc.). An error message describing password requirements is displayed. 2a. A user with the inputted email already exists in the DB. An error message indicating this is displayed. 2b. Account cannot be created due to an internal error. A "something went wrong" error is displayed.

Use case #2: Log an existing user in

#2: Log an existing user in



Use case #2: Log an existing user in

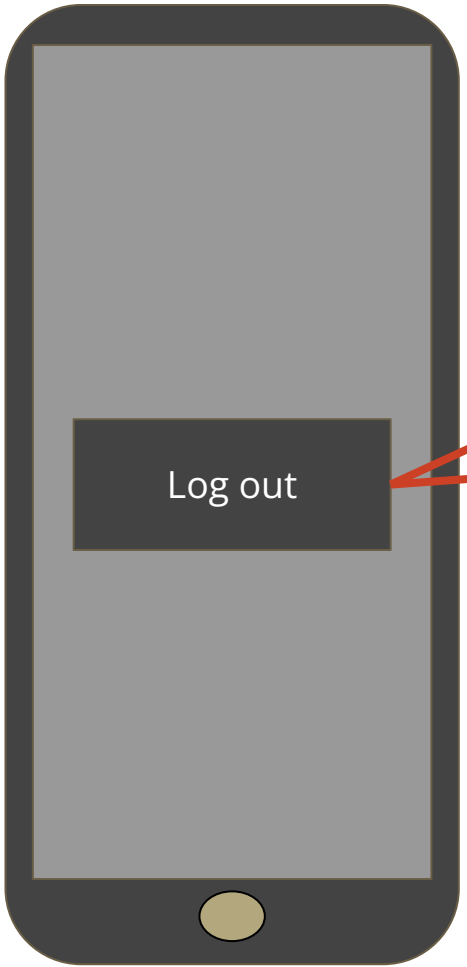
Description	Log an existing user into their account
Actors	Primary: Existing user; Secondary: DB
Goal	Log the user in
Preconditions	User is logged out and on the landing page
Postconditions	Success: User is logged in and directed to the application Failure: Login attempt fails
Exceptions	Email does not exist in the DB, password is incorrect, internal error
Trigger	User clicks “Log in” button

Use case #2: Log an existing user in

Main success scenario	Error scenarios
1. User enters their email and password 2. Application verifies credentials and saves an auth token to local storage 3. Application confirms successful login 4. User is directed into the application	2a. Email does not exist in the DB. An "account with that email or password does not exist" error message is displayed. 2b. Password is incorrect. An "account with that email or password does not exist" error message is displayed. 2c. User cannot be logged in due to an internal error. A "something went wrong" error message is displayed.

Use case #3: Log a user out of their account

#3: Log a user out of their account



Success:
User is logged out -> Redirected to landing page

Error:
Application fails to log out or redirect user

Alternative:
Auth token expires and user is automatically logged out

Use case #3: 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

Use case #3: Log a user out of their account

Main success scenario	Error scenarios	Alternative scenarios
1. Auth token is removed from local storage 2. User is directed to the landing page	1a. Logout fails. Error message indicating this is displayed. 2a. Redirect fails. User stays on same page	1. Auth token expires and the user is automatically logged out

Use case #4: Create a regatta

#4: Create a regatta

Create a regatta

ENTER

Name: _____

Timekeepers: _____

CANCEL

Success:
User clicks enter and the regatta info is saved -> Success screen is shown:

Success: Regatta created

Okay

Error:
DB can not save the regatta ->
or
Timekeeper's email is invalid ->

Error pop up is shown:

Error: Could not create regatta

Okay

Error: Timekeeper email could not be found

Okay

Use case #4: Create a regatta

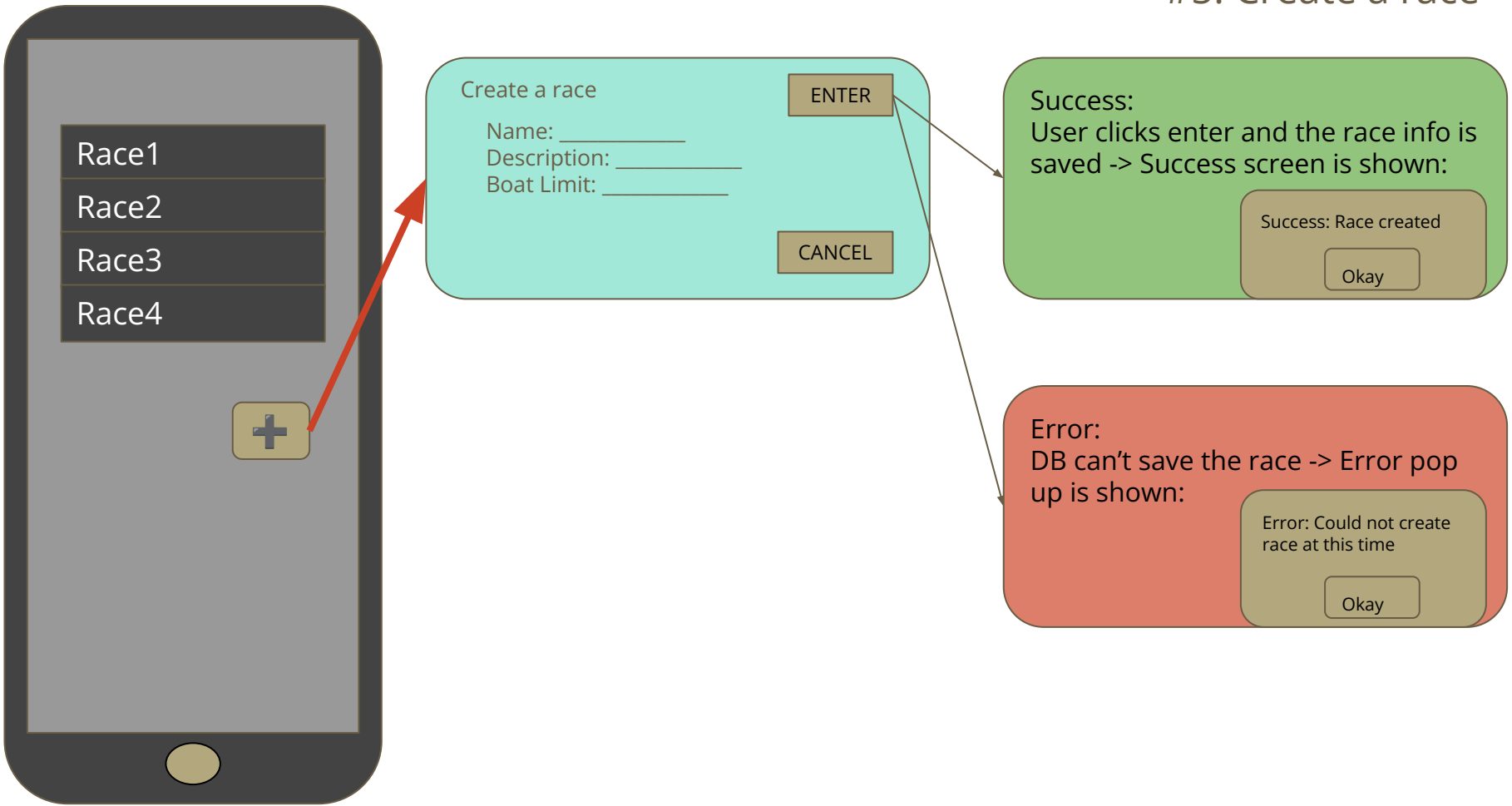
Description	Create a new regatta with the user as its admin, to which races can later be added.
Actors	Primary: User; Secondary: DB
Goal	Create a new regatta and add it to the user's "regatta list"
Preconditions	User is logged in and on the home page
Postconditions	Success: A regatta is created with 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

Use case #4: Create a regatta

Main success scenario	Error scenarios
1. User inputs the regatta name 2. User invites timekeepers by email 3. User clicks the "Save" button 4. Application saves regatta and assigns it to user in the DB 5. Application confirms successful regatta creation and creates a page for the regatta 6. User is directed to the regatta's page	2a. The inputted email does not have an associated account. An error message indicating this is displayed. 4a. The regatta cannot be saved due to an internal error. A "something went wrong" error message is displayed.

Use case #5: Create a race

#5: Create a race



Use case #5: Create a race

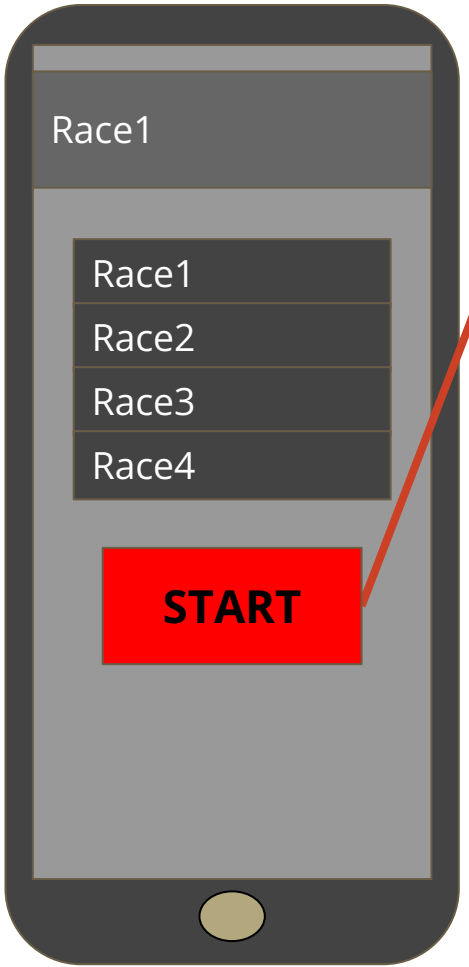
Description	Create a new race, assign it to a regatta, and add a list of registered participants
Actors	Primary: Regatta admin; Secondary: DB
Goal	To add a race that participants can add boats to and compete in, to the overall regatta.
Preconditions	User is logged in and on a regatta's page
Postconditions	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

Use case #5: Create a race

Main success scenario	Error scenarios
1. User inputs the race name 2. User enters the race description and boat limit for the race 3. User clicks the "Save" button 4. Application saves the race and adds its record in the DB 5. Application confirms successful save and creates a page for the race 6. User is directed to the race's page	1a. A race with the inputted name has already been added to the regatta's list. An error message indicating this is displayed. 4a. Race cannot be saved to to an internal error. A "something went wrong" error message is displayed.

Use case #6: Start a race

#6: Start a race



Are you sure you want to start this race?

ENTER

CANCEL

Success:
User clicks enter and the race is started -> Success screen is shown:

Success: Race started

Okay

Error:
No boats are registered for the race. ->
Another user has already started the race. ->

Error pop up is shown:

Error: A race with no boats cannot be started

Okay

Error: This race was already started

Okay

Use case #6: 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

Use case #6: Start a race

Main success scenario	Error scenarios
1. User is directed to the race's timekeeping page 2. A stopwatch begins 3. The races start time is saved to the DB	1a. No boats are registered for the race. An error message indicating this is displayed. 1b. Another user has already started the race. An error message indicating this is displayed. 2a. The stopwatch does not begin. A "something went wrong" error message is displayed. 3a. The start time is not saved to the DB. A "something went wrong" error message is displayed.

Use case #7: Mark when a boat finishes the race

#7: Mark when a boat finishes the race

Current Time:
01 : 28 : 14

Sail Number *

SAVE

CANCEL

Success:
User clicks save and the finish time is sent to the server and added to the race's page -> Success screen is shown:

Success: [###] finished at 01 : 28: 14

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
DB can't save the finish time properly -> error pop up is shown:

Could not save boat time. SMS fail.

Okay

Use case #7: 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
Goal	Save the current timestamp and boat's ID when a boat crosses the finish line
Preconditions	User is logged in, has started a race, and is on the timekeeping page
Postconditions	The boat’s finishing time and ID are saved
Exceptions	The finish time for the boat does not save correctly, this could be a SMS request fail or a DB issue
Trigger	User clicks the “Mark” button

Use case #7: Mark when a boat finishes the race

Main success scenario	Error scenarios	Alternative scenarios
1. A row is added to the "Finishing times" list with the current timestamp and an empty "ID" field 2. User fills out the ID field; selectable suggestions from the race's list of registered boats appear as they type 3. User clicks the "Save" button 4. The ID and timestamp are sent to the server via SMS and added to the race's page	1. Info cannot be sent due to an internal error. Application periodically reattempts request until it is sent. 2. Info could not be saved in the database	1. The inputted ID does not exist in the list. User manually finishes typing it in. 2. 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

Use case #8: Disqualify a boat from a race

#8: Disqualify a boat from a race

Disqualify a boat:

Sail Number *

SAVE

v

DNS DNF

CANCEL

Success:
User inputs sail number, selects DNF or DNS then clicks save. The boat is removed from the race.

Success: [###] has been disqualified.

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
SMS request failed -> error pop up is shown:

Could not disqualify boat. SMS fail.

Okay

Use case #8: 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	The boat is disqualified from the race, or it remains in the running
Exceptions	Specified boat ID is not in the race, SMS request fails
Trigger	User clicks the "Add DNS/DNF" button

Use case #8: Disqualify a boat from a race

Main success scenario	Error scenarios
1. User inputs the boat ID and selects whether it was a DNS or DNF 2. User clicks the "Save" button 3. The ID and result are sent to the server via SMS and added to the race's page	1. Specified ID is not registered for the race. An error message indicating this is displayed. 2. Info cannot be sent due to an internal error. Application periodically reattempts request until it is sent.

Use case #9: Edit a regatta

#9: Edit a regatta

EDIT REGATTA

ENTER

New Name: _____

New Date: _____

CANCEL

Success:
User clicks enter and the boats information is changed -> Success screen is shown:

Success: Regatta edited

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
DB can't update the regatta -> error pop up is shown:

Could not update regatta

Okay

Use case #9: Edit a regatta

Description	A regatta admin can edit the regatta name and date
Actors	Primary: Regatta admin, Secondary: DB
Goal	Change the regatta name or date of occurence
Preconditions	User is logged in, has created a regatta, and is on that regatta page
Postconditions	The regatta is successfully edited or the application fails to edit the data
Exceptions	DB error when updating information
Trigger	User clicks the “Edit” button

Use case #9: Edit a regatta

Main success scenario	Error scenarios	Alternative scenarios
1. User inputs the new name or date into the provided field if applicable 2. User presses "Save" button 3. Application saves the new info to the DB 4. Application confirms successful change	1. The name is not saved to the DB. A "something went wrong" error message is displayed.	1. User clicks the "Cancel" button. Edits are not saved.

Use case #10: Delete a regatta

#10: Delete a regatta

DELETE REGATTA

CONFIRM

Are you sure you want to delete this regatta?

CANCEL

Success:
User clicks confirm and the regatta is deleted. -> Success screen is shown:

Success: Regatta deleted.

Okay

Alternate: User Clicks cancel:
The regatta is not deleted.

Error:
DB can't delete the regatta -> error pop up is shown:

Could not delete regatta.

Okay

Use case #10: 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 delete from the DB.
Exceptions	DB error.
Trigger	The user clicks on the "Delete Regatta" button.

Use case #10: Delete a regatta

Main success scenario	Error scenarios	Alternative scenarios
1. A popup confirming that the user wants to delete the regatta and all its races is displayed. 2. User clicks the “Confirm” button. 3. Application removes the regatta from the DB. 4. Application confirms successful deletion of the regatta.	3a. The regatta is not removed from the DB properly. A “something went wrong” error message is displayed.	2. The user does not want to delete the regatta and clicks the “Cancel” button. The regatta is not deleted.

Use case #11: Invite a timekeeper to a regatta

#11: Invite a timekeeper to a regatta

INVITE TIMEKEEPERS

INVITE

New Timekeepers Emails:

CANCEL

Success:
User clicks invite and the user is invited. -> Success screen is shown:

Success: Invite sent.

Okay

Alternate: User Clicks cancel:
No other users are invited to join as a timekeeper.

Error:
- User email does not exist. -> error message is shown.
- Application fails to grant access. - error message is shown.
- Application fails to notify user of success (nothing is shown).

One or more emails do not exist.

Okay

Failed to invite user(s).

Okay

Use case #11: Invite a timekeeper to a regatta

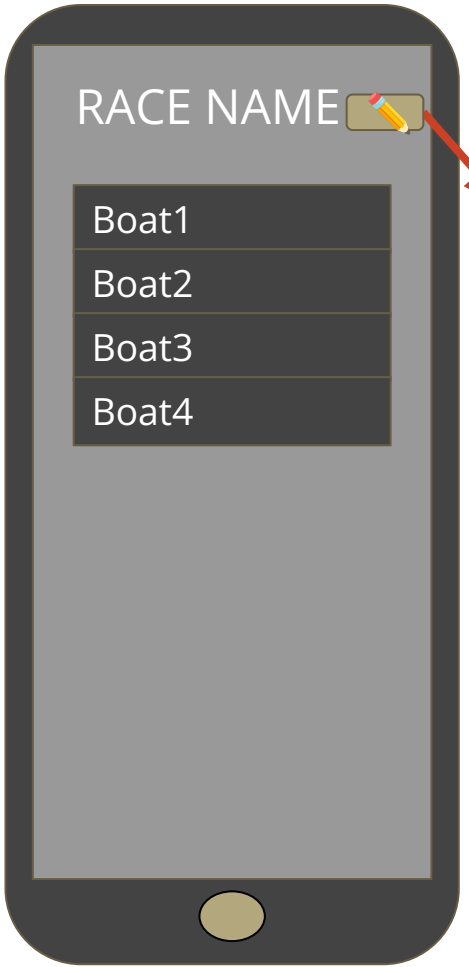
Description	A regatta's admin can invite other users by email to act as a timekeeper for races in the regatta.
Actors	Primary: Regatta admin; Secondary: DB
Goal	Invite another user to timekeep for a regatta by sending an email.
Preconditions	User is logged in, has created a regatta, and is on that regatta's 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.

Use case #11: Invite a timekeeper to a regatta

Main success scenario	Error scenarios	Alternative scenarios
1. User inputs one or more emails and invites the people associated with the emails. 2. Application grants users associated with those emails access to the regatta's timekeeping features. 3. Application notifies user and invited users of success.	1a. User associated with an inputted email doesn't exist. An error message indicating this is displayed. 2a. Application fails to grant the invitees access. An error message indicating this is displayed. 3a. Application fails to notify user and invitees of success.	1. User clicks the "Cancel" button. Edits are not saved.

Use case #12: Edit a race's name

#12: Edit a race's name



EDIT RACE

New Name: _____

ENTER

CANCEL

Success:
User clicks enter and the boats information is changed -> Success screen is shown:

Success: Race name changed.

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
DB can't update the boat -> error pop up is shown:

Could not update Race.

Okay

Use case #12: Edit a race's name

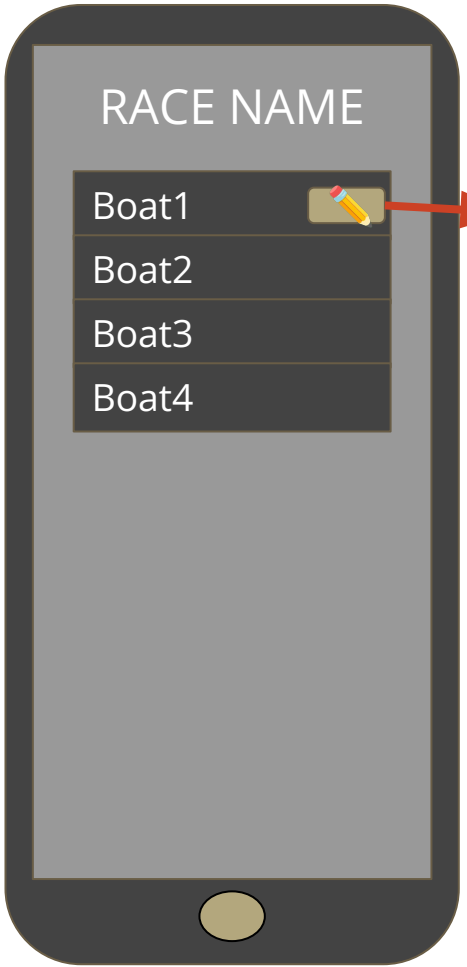
Description	A regatta's admin can change the race's name.
Actors	Primary: Regatta admin; Secondary: DB.
Goal	Rename a race.
Preconditions	User is logged in, has created a regatta, and is on the page of a race that belongs to the regatta
Postconditions	Success: The race is successfully renamed. Failure: Application fails to rename the race.
Exceptions	Invalid input, DB error.
Trigger	User clicks the "Edit" button.

Use case #12: Edit a race's name

Main success scenario	Error scenarios	Alternative scenarios
1. User types a new name in. 2. User clicks the “Save” button. 3. Application saves new name to the DB. 4. Application confirms successful name change.	1a. Invalid Input - An error message indicating name requirements is displayed. 3a. Name is not saved to the DB - a “something went wrong” error is displayed.	2. User clicks the “Cancel” button. Edits are not saved.

Use case #13: Edit a boat in a race

#13: Edit a boat in a race



EDIT BOAT

ENTER

New ID: _____

New Name: _____

New Time: _____

CANCEL

Success:
User clicks enter and the boats information is changed -> Success screen is shown:

Success: Boat edited

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
DB can't update the boat -> error pop up is shown:

Could not update Boat

Okay

Use case #13: Edit a boat in a race

Description	Allows admins to change a given boat’s information
Actors	Regatta admin, timekeeper
Goal	User edits a registered boat's ID, participants' names, or finishing time
Preconditions	User is logged in, has created a regatta, and is on the page of a race that belongs to the regatta
Postconditions	A existing boat’s information is changed
Exceptions	User inputs invalid data or the program is unable to update the information
Trigger	The user clicks the "Edit" button

Use case #13: Edit a boat in a race

Main success scenario	Error scenarios	Alternative scenarios
1. The user edits the attribute that they wish to change (the ID, the name, and/or the finishing time) 2. The user clicks the "Save changes" button to save the information 3. Application updates the DB 4. Application notifies the user of succes	3a. The application fails to update the information properly. A "something went wrong" error message is displayed. 4a. The application fails to let the user know it succeeded.	2. User clicks the "Cancel" button. Edits are not saved.

Use case #14: Add a boat to a race

#14: Add a boat to a race

ADD BOAT

ENTER

ID: _____

Name: _____

CANCEL

Success:
User clicks enter and the boats information is changed -> Success screen is shown:

Success: Boat Added

Okay

Alternate: User Clicks cancel: nothing is changed

Error:
DB can't add the boat -> error pop up is shown:

Could not Add Boat

Okay

Use case #14: Add a boat to a race

Description	Adds a boat new boat to a race
Actors	Admin, Timekeeper
Goal	The user adds a new boat to a race
Preconditions	User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page
Postconditions	A new boat is added
Exceptions	A boat is invalid or DB error
Trigger	User Clicks the "Add boat" button

Use case #14: Add a boat to a race

Main success scenario	Error scenarios
1. User inputs the boat's name and its participants' names 2. Application saves the new boat to the DB 3. Application confirms success and the new boat is added to the list of boats in that particular race	1a. The user’s boat ID already exists in the race. An error message indicating this is displayed. 2a. The new boat is not saved. A "something went wrong" error is displayed.

Use case #15: Delete a boat from a race

#15: Delete a boat from a race

ARE YOU SURE YOU WANT TO DELETE BOAT 1?

Yes

No

Success:
User clicks enter and the boat is deleted-> Success screen is shown:

Success: Boat deleted

Okay

Alternate: User Clicks no: nothing is changed

Error:
DB can't update the boat -> error pop up is shown:

Could not delete Boat

Okay

Use case #15: Delete a boat from a race

Description	Removes a boat from race
Actors	Regatta admin, timekeeper
Goal	Removes a boat from a race
Preconditions	User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's page
Postconditions	Boat is deleted
Exceptions	DB error
Trigger	User clicks the delete button next to a boat

Use case #15: Delete a boat from a race

Main success scenario	Error scenarios	Alternative scenarios
1. A popup confirming that the user wants to delete the boat is displayed 2. User clicks the "Confirm" button 3. Application removes the boat from the DB. 4. Application confirms successful deletion of the boat	3a. The regatta is not removed from the DB properly. A “something went wrong” error message is displayed.	2. The user does not want to delete the boat and clicks the "Cancel" button. The boat is not deleted.

Use case #16: Delete a race

#16: Delete a race

ARE YOU SURE YOU WANT TO DELETE THIS RACE?

CONFIRM

CANCEL

Success:
User clicks enter and the race is deleted-> Success screen is shown:

Success: Race has been deleted.

Okay

Alternate:
- User Clicks cancel: nothing is changed.
- There is a race in progress: application ends race and active users are kicked.

Error:
DB can't delete the race -> error pop up is shown:

Could not delete Race.

Okay

Use case #16: Delete a race

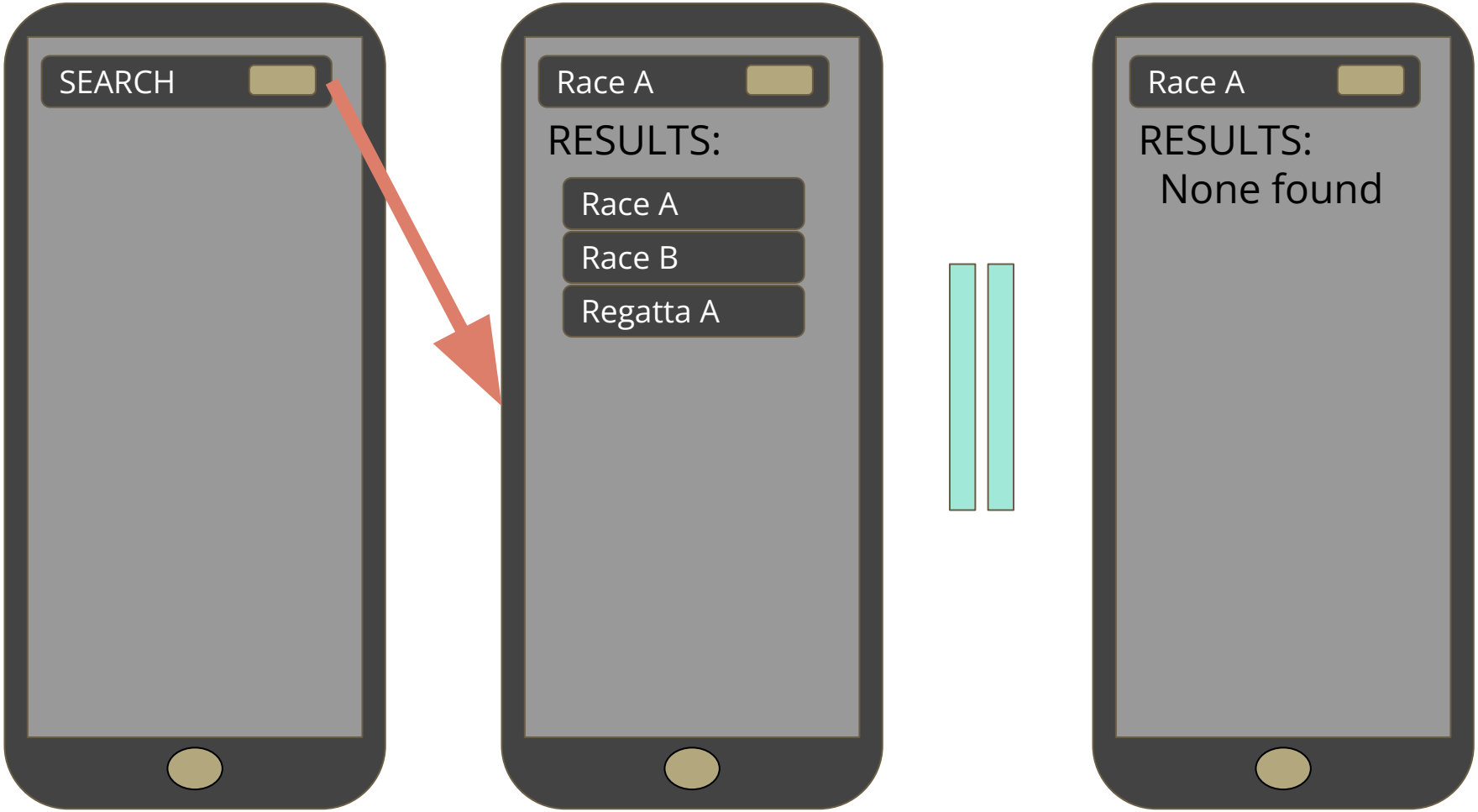
Description	A regatta's admin can delete a race, removing it from the DB and regatta
Actors	Primary: Regatta admin; Secondary: Database.
Goal	User deletes the race from both the DB and the regatta.
Preconditions	User is logged in, is a regatta admin or timekeeper for an existing race, and is on the race's 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.

Use case #16: Delete a race

Main success scenario	Error scenarios	Alternative scenarios
1. A popup confirming that the user wants to delete the race is displayed 2. User clicks the "Confirm" button 3. Application removes the race from the DB. 4. Application confirms successful deletion of the race	3a. Race is not removed from the DB. A "something went wrong" error message is displayed.	1. The race is in progress. The message confirms that the user wants to delete the race while it's in progress. 3. The race is in progress. Application ends the race and timekeepers are kicked out.

Use case #17: Search for regattas and races

#17: Search for regattas and races



Use case #17: Search for regattas and races

Description	Search for a regatta based on race or regatta name
Actors	All stakeholders
Goal	Search for current or past regattas
Preconditions	User is on the landing page and types into the search bar
Postconditions	User sees a list of relevant regattas
Exceptions	DB error or no matching regattas exist
Trigger	User types into the search bar

Use case #17: Search for regattas and races

Main success scenario	Error scenarios	Alternative scenarios
1. User inputs their search query (regatta name / race name / participant name / boat ID) and clicks "Search" or enter button 2. Application queries DB to find matching results 3. A list of results is displayed	2a. DB query fails due to an internal error. A "something went wrong" error message is displayed.	3. No matching results. A message indicating this is displayed.

Thank you!