

# Event Tracker 1.0

SPECIFICATIES

EVERT DE SCHRYVER

## INHOUDSTABEL

Probleemdomein .....	3
Omschrijving Probleemdomein .....	3
Authenticatie & Authorisatie .....	4
Procesdiagram .....	5
Use Case Diagram .....	6
Use Cases Overzicht .....	7
Activity Diagrams .....	8
Login .....	8
Reset Password .....	9
Change Password .....	10
View Upcoming Events .....	11
View My Events .....	11
(Un)subscribe .....	12
Add Event .....	13
Edit Event .....	14
Cancel Event .....	15
Delete Event .....	16
View All Users .....	17
Add User .....	18
Edit User .....	19
Delete User .....	20
User Interface .....	21
User Interface Mockups .....	21
Login .....	21
Forgot Password .....	22
Reset Password .....	23
Change Password .....	24
Upcoming Events .....	25
Add Event .....	26
Edit Event .....	27
Users Overview .....	28
Add User .....	29
Edit User .....	30
Storyboards .....	31
Login .....	31

Forgot Password .....	32
Change Password .....	33
Sidebar Navigation .....	34
Event Management .....	35
User Management .....	36
Ontwikkelingstack .....	37
Doelplatform .....	37
Technologieën .....	37
Front end .....	37
Back end .....	37
Tools .....	37
Architectuur .....	38
Application Layers .....	38
Data Access Layer .....	38
De Models Layer .....	39
De Services Layer .....	40
De Web Application Layer .....	40
Class Diagram .....	43
ErrorController .....	43
AccountController .....	44
EventsController .....	45
UserProfilesController .....	46
Database Diagram .....	47
Technologische Uiteenzettingen .....	48
Mobile First .....	48
Extensions .....	51
Alerts .....	51
Validators .....	53
Vraagstukken Tijdens Development .....	54
Timeframes toevoegen .....	54
Overposting .....	58

## PROBLEEMDOMEIN

### OMSCHRIJVING PROBLEEMDOMEIN

---

EventTracker is een responsive, 'mobile-first' webapplicatie waarmee een gebruiker zich kan inschrijven op diverse evenementen. Door zich in te schrijven op een evenement, geven zij aan te zullen participeren in de organisatie van dit evenement. Afhankelijk van hun niveau van autorisatie kunnen bepaalde gebruikers evenementen creëren, beheren, annuleren en/of verwijderen.

Een evenement beschikt over volgende zaken:

- Een naam
- Één of meerdere tijdspannes bestaande uit een datum met daaraan een start- en einduur gekoppeld
- Een locatie bestaande uit een stad en een provincie
- Een korte omschrijving
- Een minimum aantal gewenste participanten
- Een lijst van reeds ingeschreven gebruikers

Gebruikers moeten zich inloggen alvorens ze gebruik kunnen maken van de applicatie. Zij kunnen ook aangeven of ze wensen dat de applicatie hun gebruikersprofiel via een cookie onthoudt. Hierdoor kunnen zij het inlog proces in de toekomst vermijden. Zelf kan men geen gebruikersprofiel aanmaken. Dit gebeurt door een administrator. Eens een nieuw gebruikersprofiel is aangemaakt, stuurt de applicatie een confirmatiemail naar het email adres dat gebruikt is tijdens het aanmaken van de gebruiker in kwestie. Een gebruiker kan zich niet inloggen zolang hij zijn profiel niet heeft geactiveerd door de instructies in de confirmatiemail te volgen.

Een gebruikersprofiel beschikt over volgende zaken:

- Voornaam
- Familienaam
- Email
- Paswoord
- Gebruikersrol

Een gebruiker krijgt na het inloggen een overzicht te zien van alle toekomstige evenementen. Evenementen uit het verleden worden niet weergegeven door de applicatie. Alle gebruikers kunnen zich voor elk getoond evenement in- of uitschrijven tenzij het evenement geannuleerd is. Er staat geen maximum op het aantal inschrijvingen. De applicatie zal wel visueel kenbaar maken wanneer het gewenste aantal participanten is bereikt. Een gebruiker kan ook een overzicht krijgen van alle evenementen waarop hij zich heeft ingeschreven.

Een evenement kan door een superuser geannuleerd worden. Wanneer dit het geval is, zal de applicatie ook dit visueel kenbaar maken. Een gebruiker kan zich in dit geval niet meer inschrijven voor het evenement. Ook kan een superuser de details van een geannuleerd evenement niet langer aanpassen of het evenement verwijderen. Een geannuleerd evenement kan door een superuser wel terug beschikbaar worden gesteld. Eens dit is gebeurd, kunnen alle functionaliteiten terug worden toegepast.

Een verder overzicht van de verschillende gebruikersrollen met de daaraan gekoppelde autorisatie kan samen met een diepere omschrijving van het authenticatieproces gevonden worden in het hoofdstuk '[Authenticatie & Autorisatie](#)'.

## AUTHENTICATIE & AUTHORISATIE

---

Een gebruikersprofiel kan alleen worden aangemaakt door een administrator. Een voor de applicatie ongekende gebruiker kan zelf geen gebruikersprofiel aanmaken. Iemand die toegang wenst tot applicatie moet een aanvraag indienen bij een administrator. Hierbij geeft men een geldig emailadres dat men wenst te koppelen aan het gebruikersprofiel. Dit emailadres wordt gebruikt als unieke gebruikersnaam. Dit betekent dat de applicatie geen nieuw gebruikersprofiel zal aanvaarden waarvan het opgegeven emailadres reeds gekoppeld is aan een bestaand gebruikersprofiel.

De administrator voorziet de nieuwe gebruiker van een paswoord. Dit paswoord is onderworpen aan volgende vereisten:

- Minimum 8 karakters
- Minimum 1 uppercase karakter
- Minimum 1 lowercase karakter
- Minimum 1 cijfer
- Minimum 1 punctuatie- of symboolkarakter

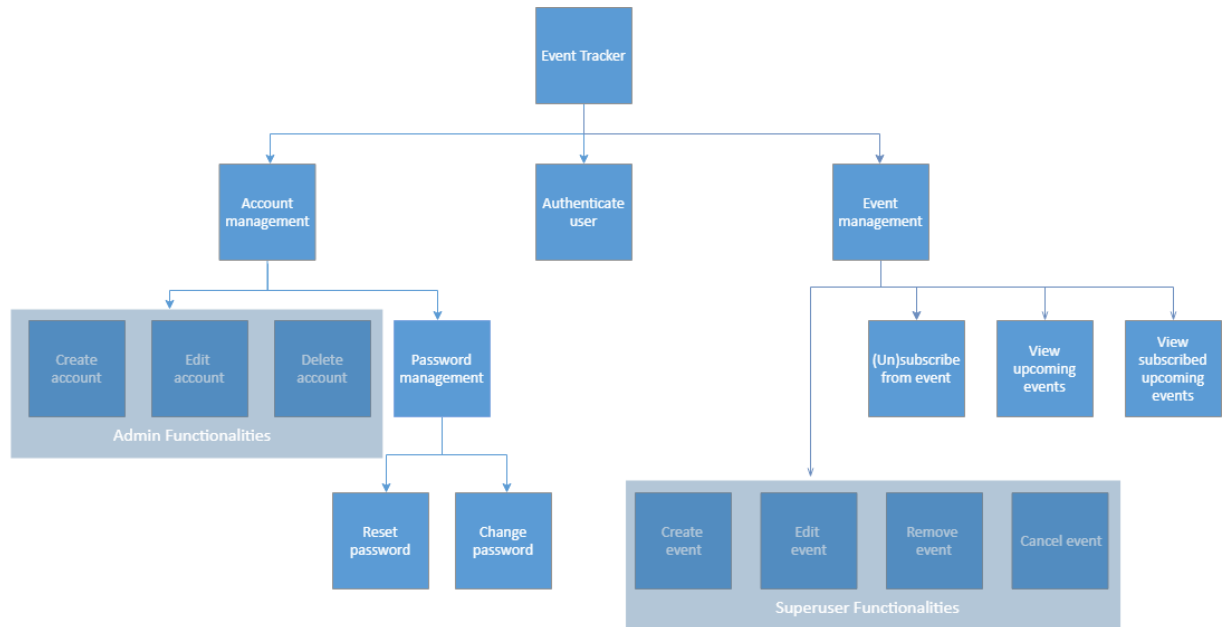
Eens een gebruikersprofiel is aangemaakt, zal de applicatie een confirmatiemail sturen naar het opgegeven email adres. Het doel van deze mail is om er zeker van te zijn dat het opgegeven emailadres daadwerkelijk bestaat. De mail bevat een confirmatielink. Wanneer de gebruiker op deze link klikt, zal zijn gebruikersprofiel geactiveerd zijn. De link is 24 uur geldig, hierna vervalt hij en kan hij bijgevolg niet meer gebruikt worden om het gebruikersprofiel te activeren. Een gebruiker kan zich pas inloggen indien zijn gebruikersprofiel geactiveerd is.

Een administrator voorziet een gebruiker tenslotte van een gebruikersrol. Volgende rollen kunnen worden toegewezen:

- Basic user:
  - Zij kunnen zich inloggen
  - Zij kunnen een overzicht krijgen van alle toekomstige evenementen
  - Zij kunnen een overzicht krijgen van alle toekomstige evenementen waarop zij zich hebben ingeschreven
  - Zij kunnen zich in- of uitschrijven voor evenementen
  - Zij kunnen het paswoord gelinkt aan hun profiel wijzigen of resetten
- Superuser:
  - Zij kunnen alles wat een basic user kan
  - Zij kunnen evenementen creëren, aanpassen, annuleren en verwijderen
  - Zij kunnen voor elk evenement deelnemers verwijderen
- Administrator:
  - Zij kunnen alles wat superusers kunnen
  - Zij kunnen gebruikersprofielen creëren, aanpassen en verwijderen

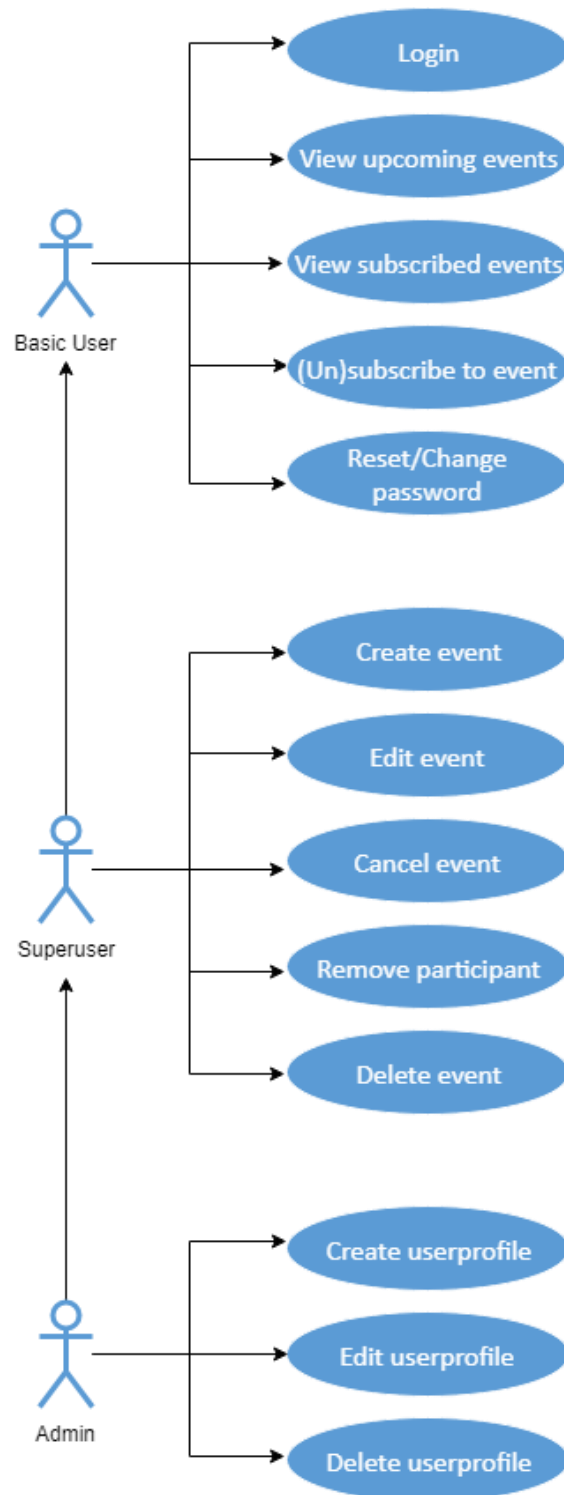
## PROCESDIAGRAM

Onderstaand diagram schets een beeld van alle processen die plaats vinden binnen de context van de applicatie.



## USE CASE DIAGRAM

Onderstaand diagram schets een beeld van alle use cases die plaats vinden binnen de context van de applicatie:



## USE CASES OVERZICHT

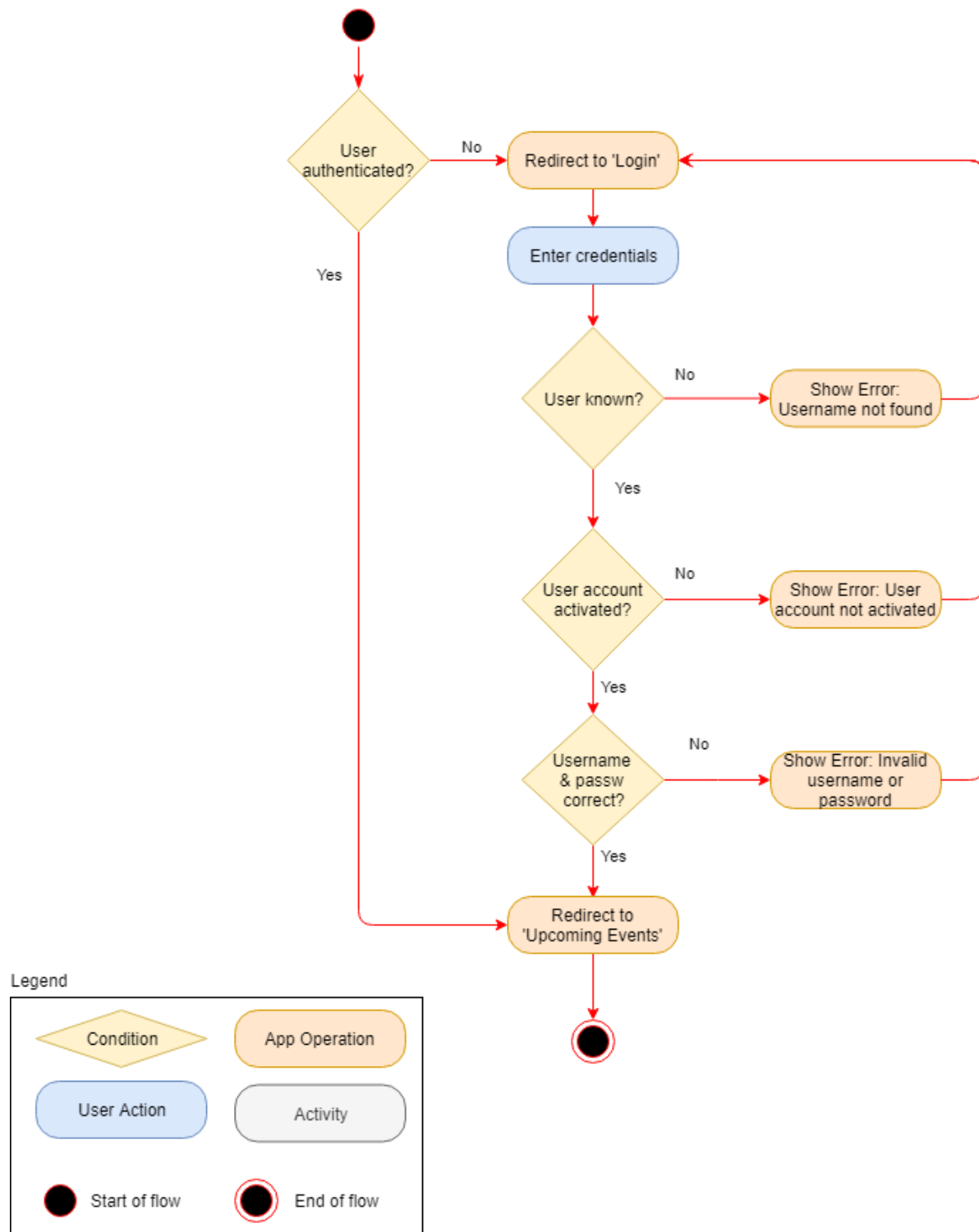
Onderstaand overzicht schets een beeld van alle use cases die plaats vinden binnen de context van de applicatie.

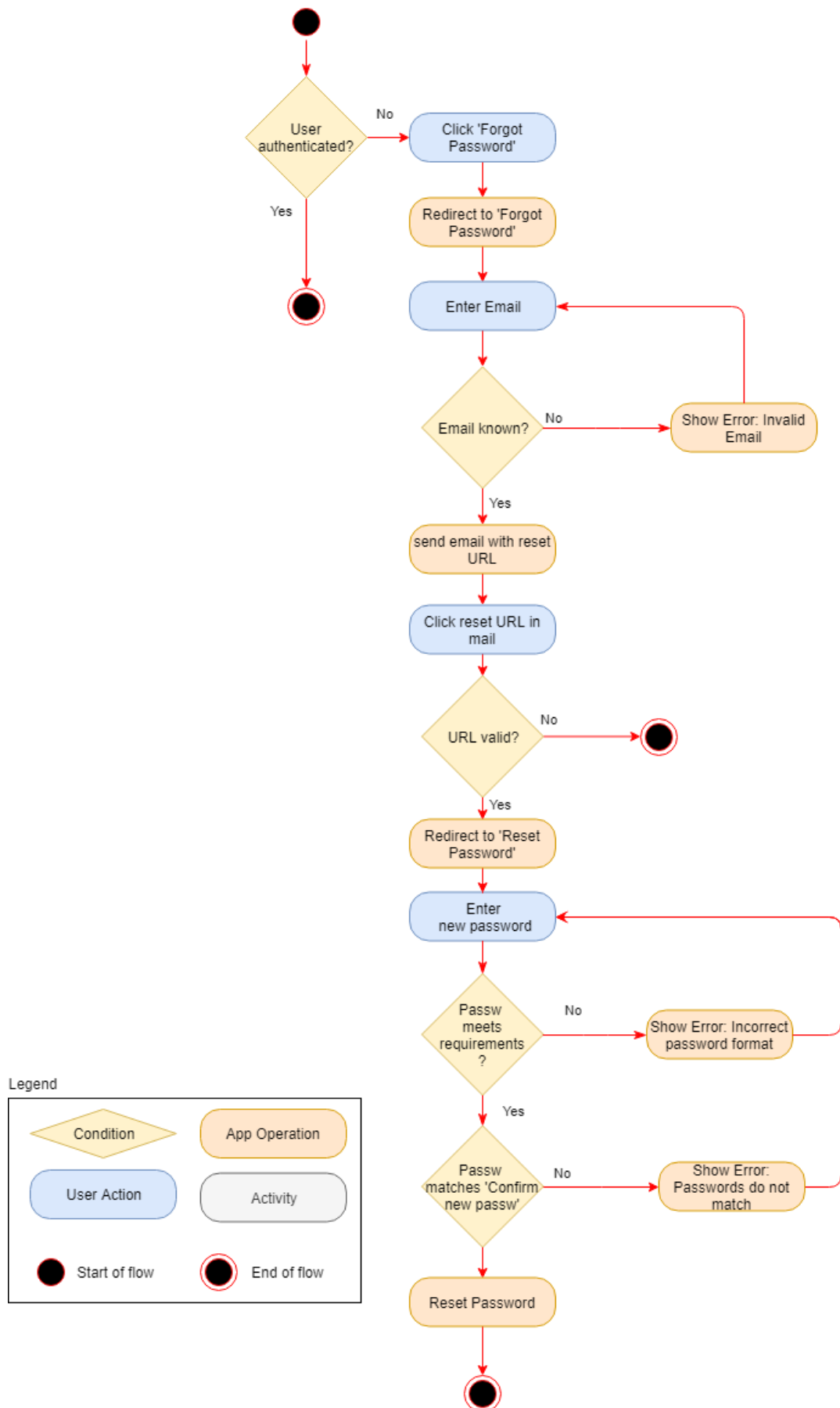
Als een...	Zou ik graag...	Zodat ik...
Basic user	Inloggen	Gebruik kan maken van de applicatie
Basic user	Een lijst krijgen van alle toekomstige evenementen	Kan zien welke evenementen er gepland staan
Basic user	Een lijst krijgen van alle toekomstige evenementen waarvoor ik me heb ingeschreven	Een overzicht heb van alle toekomstige evenementen waar mijn aanwezigheid wordt verwacht
Basic user	Mezelf inschrijven voor een evenement	Mijn aanwezigheid kenbaar kan maken
Basic user	Mezelf uitschrijven voor een evenement	Mijn afwezigheid kenbaar kan maken
Basic user	Mijn paswoord wijzigen	Goede beveiligingsnormen kan hanteren
Basic user	Mijn paswoord resetten	Een nieuw paswoord kan instellen indien ik het vergeten ben
Superuser	Evenementen aanmaken	Nieuwe evenementen beschikbaar kan stellen voor gebruikers
Superuser	Evenementen aanpassen	De praktische info van een evenement kan wijzigen
Superuser	Evenementen verwijderen	Onnodige evenementen kan verwijderen
Superuser	Evenementen annuleren	Gebruikers op de hoogte kan stellen wanneer een evenement niet doorgaat
Admin	Gebruikersprofielen aanmaken	Aan gebruikers de mogelijkheid geef gebruik te maken van de applicatie
Admin	Gebruikersprofielen verwijderen	Bepaalde gebruikers de toegang tot de applicatie kan ontfeggen
Admin	Gebruikersprofielen aanpassen	De gebruikersrol, email of naam van een gebruiker kan wijzigen

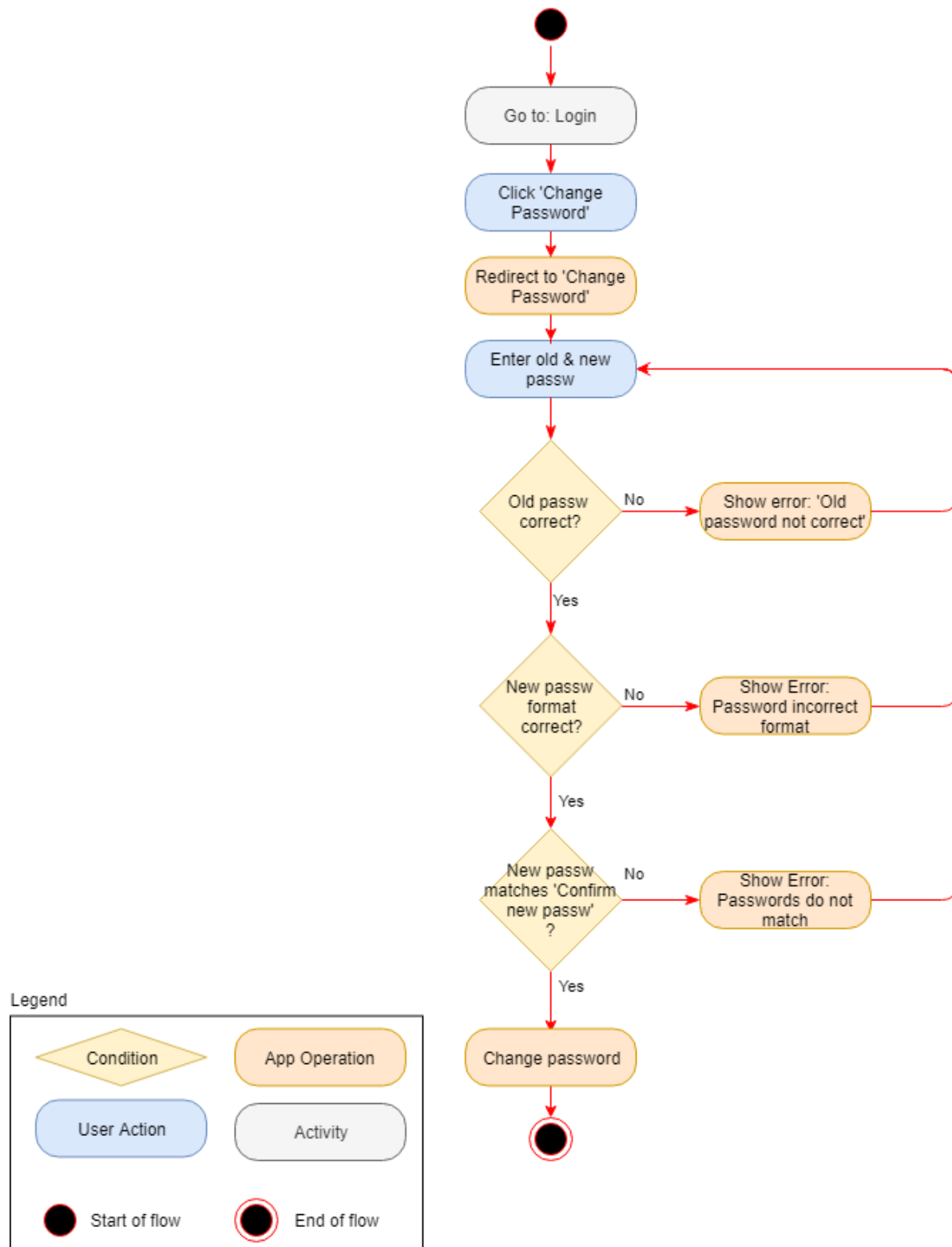


## ACTIVITY DIAGRAMS

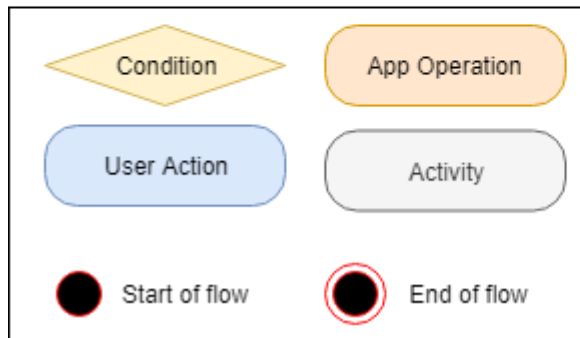
### LOGIN



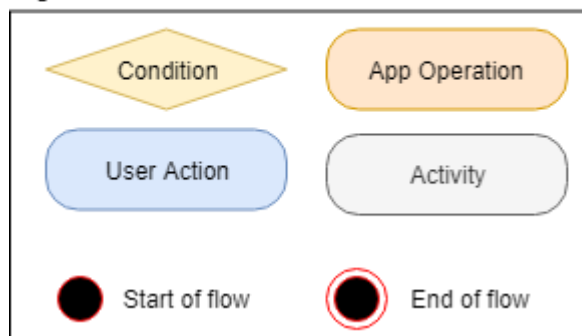


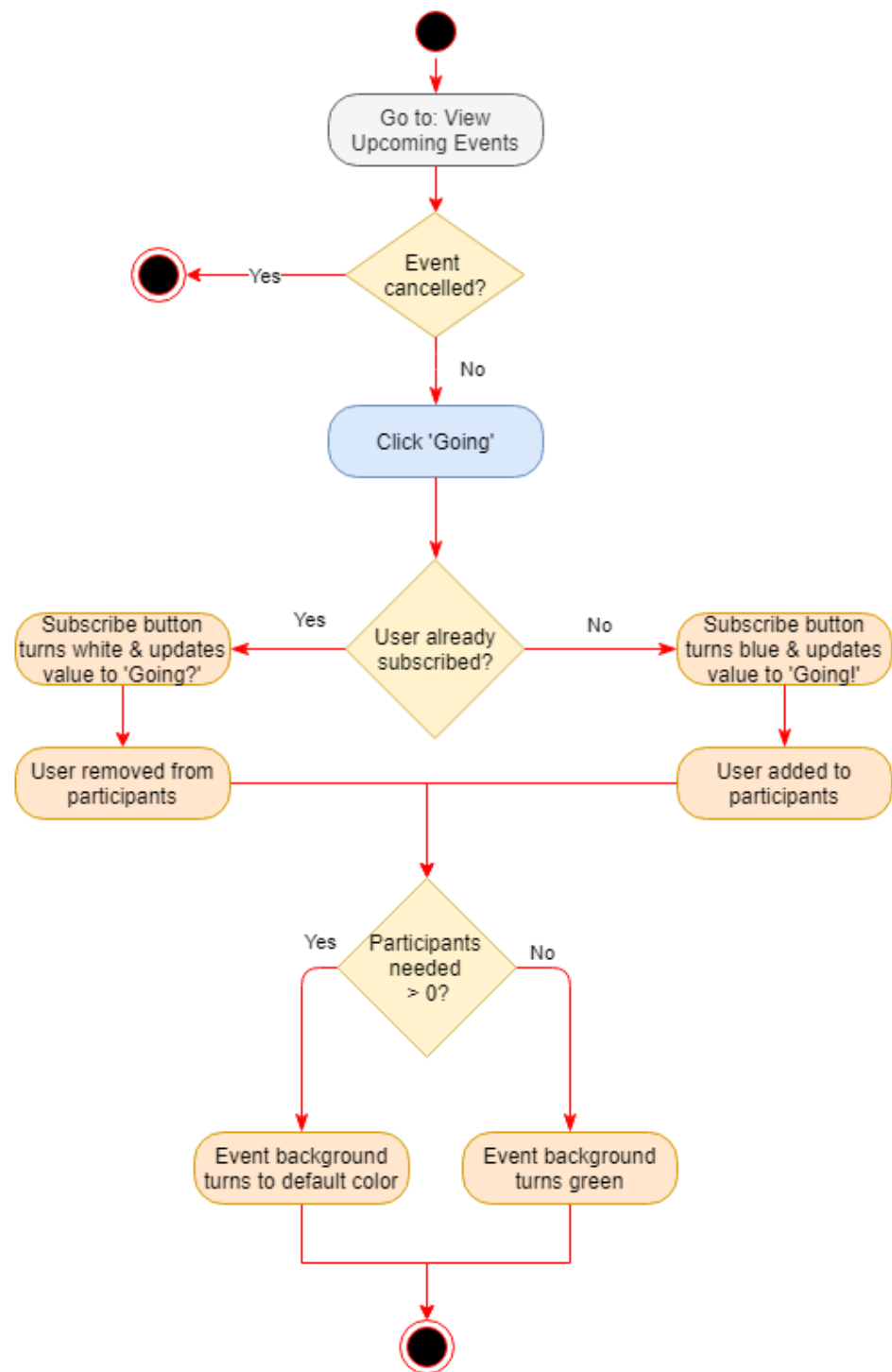


## Legend

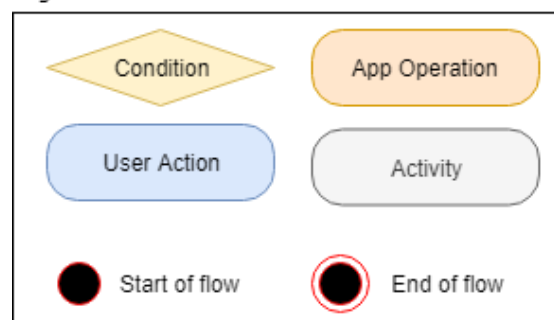


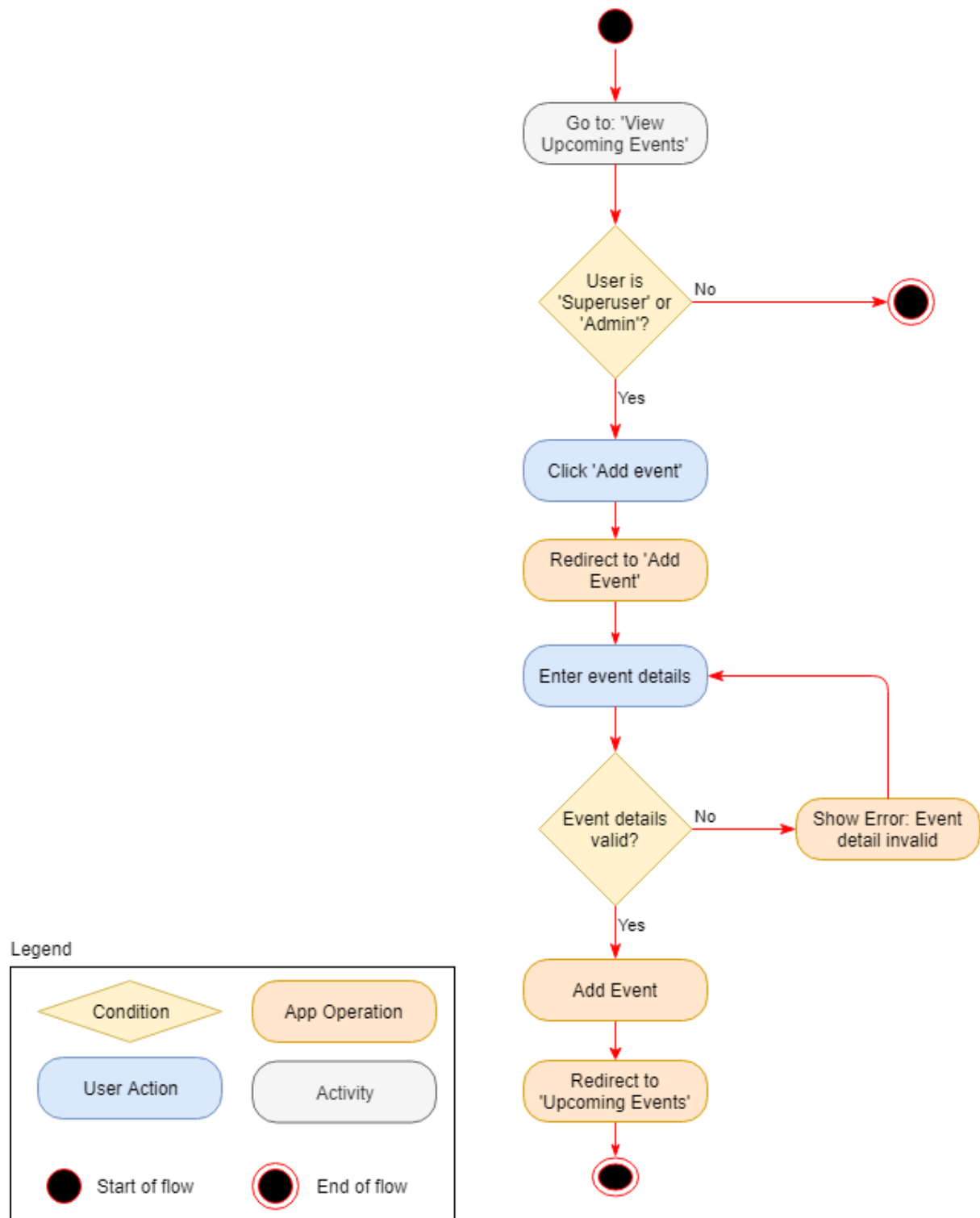
## Legend

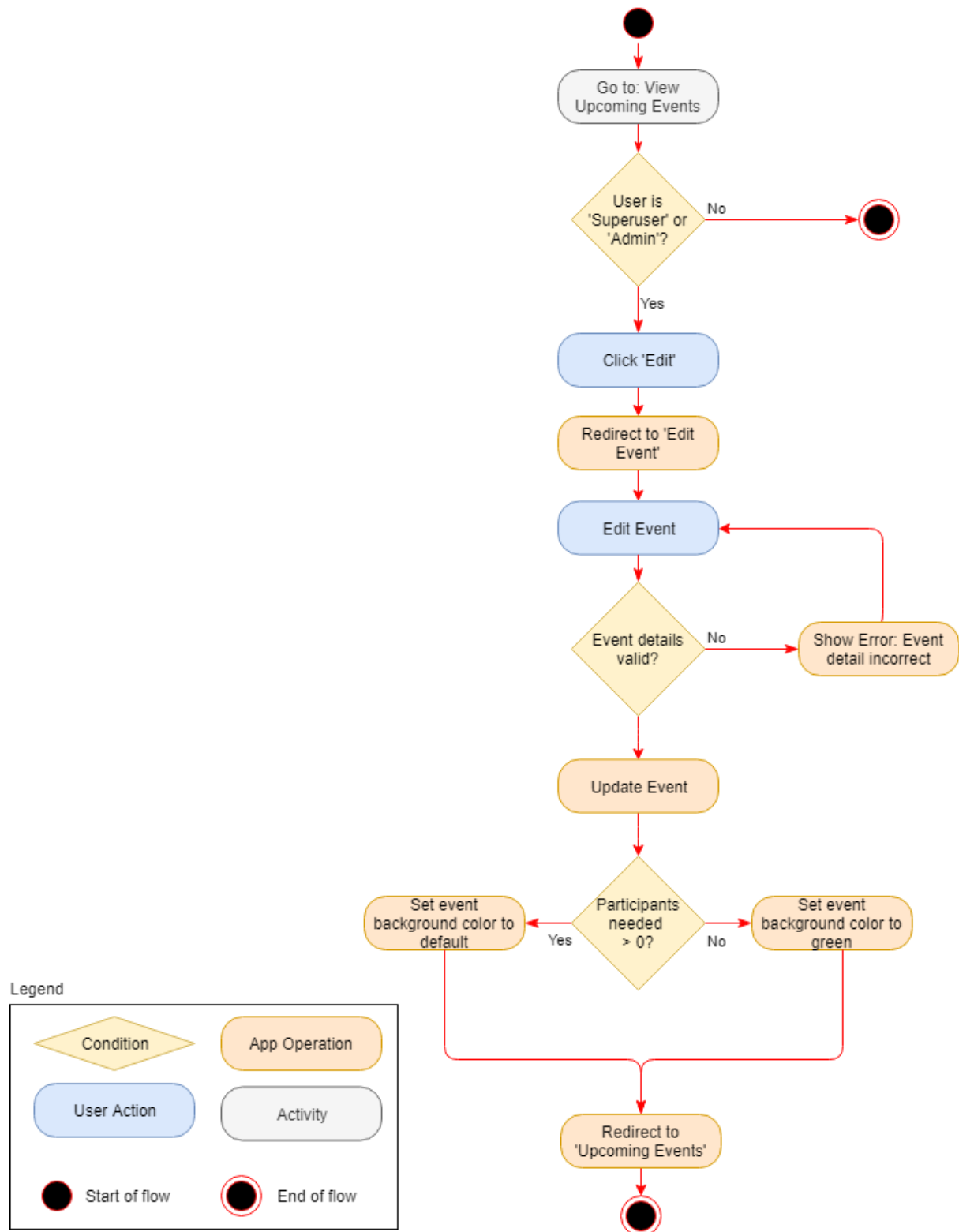


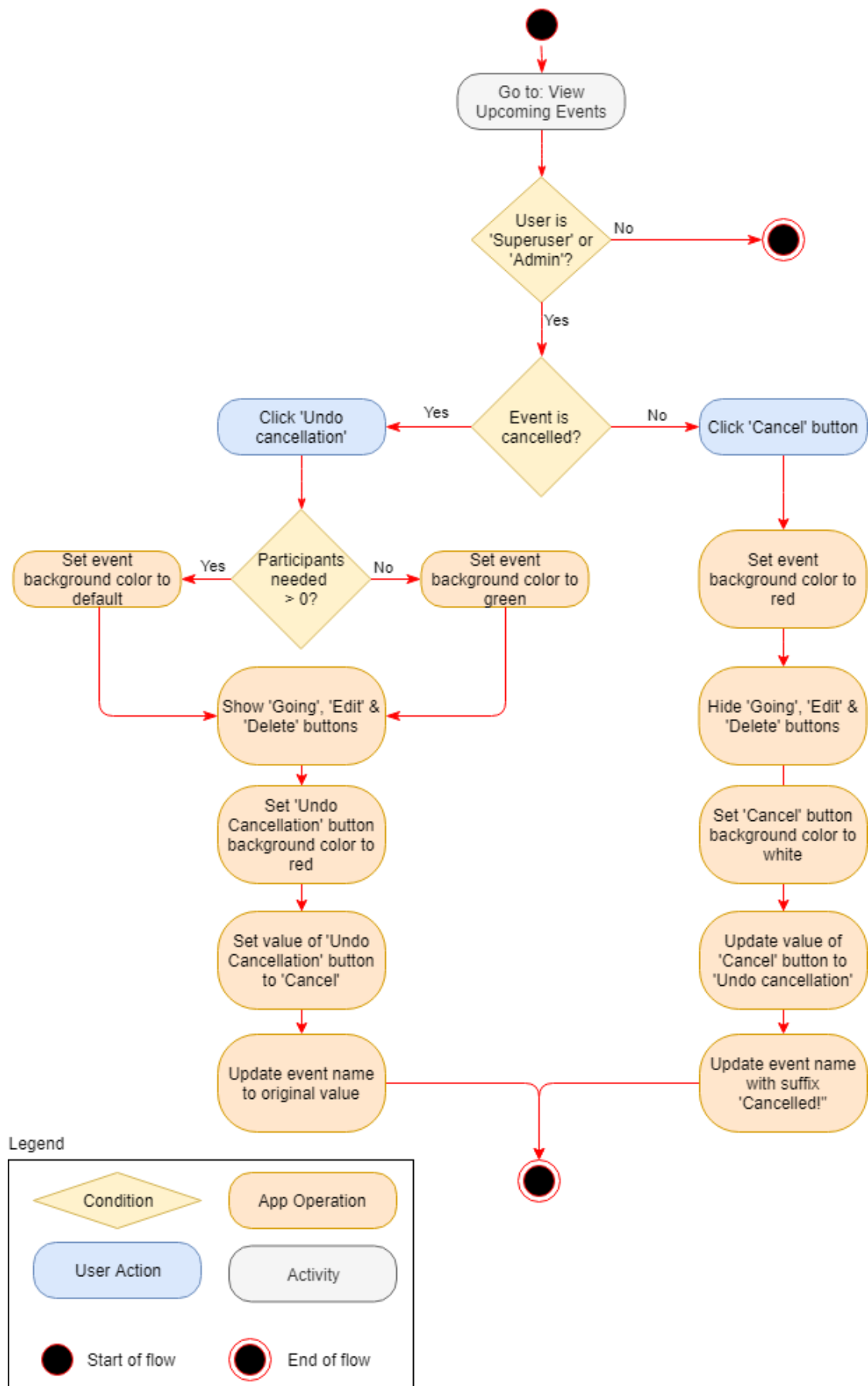


## Legend

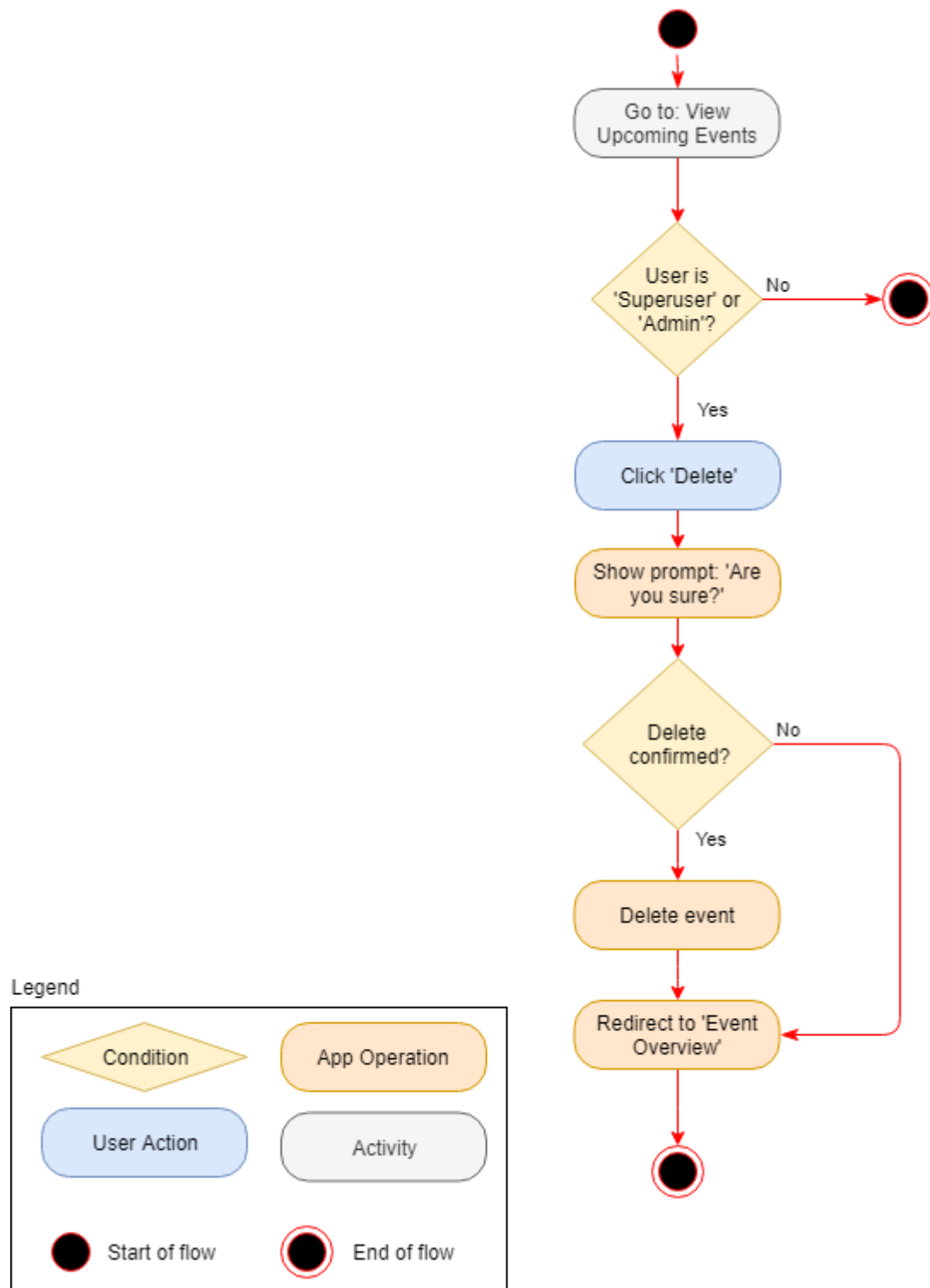


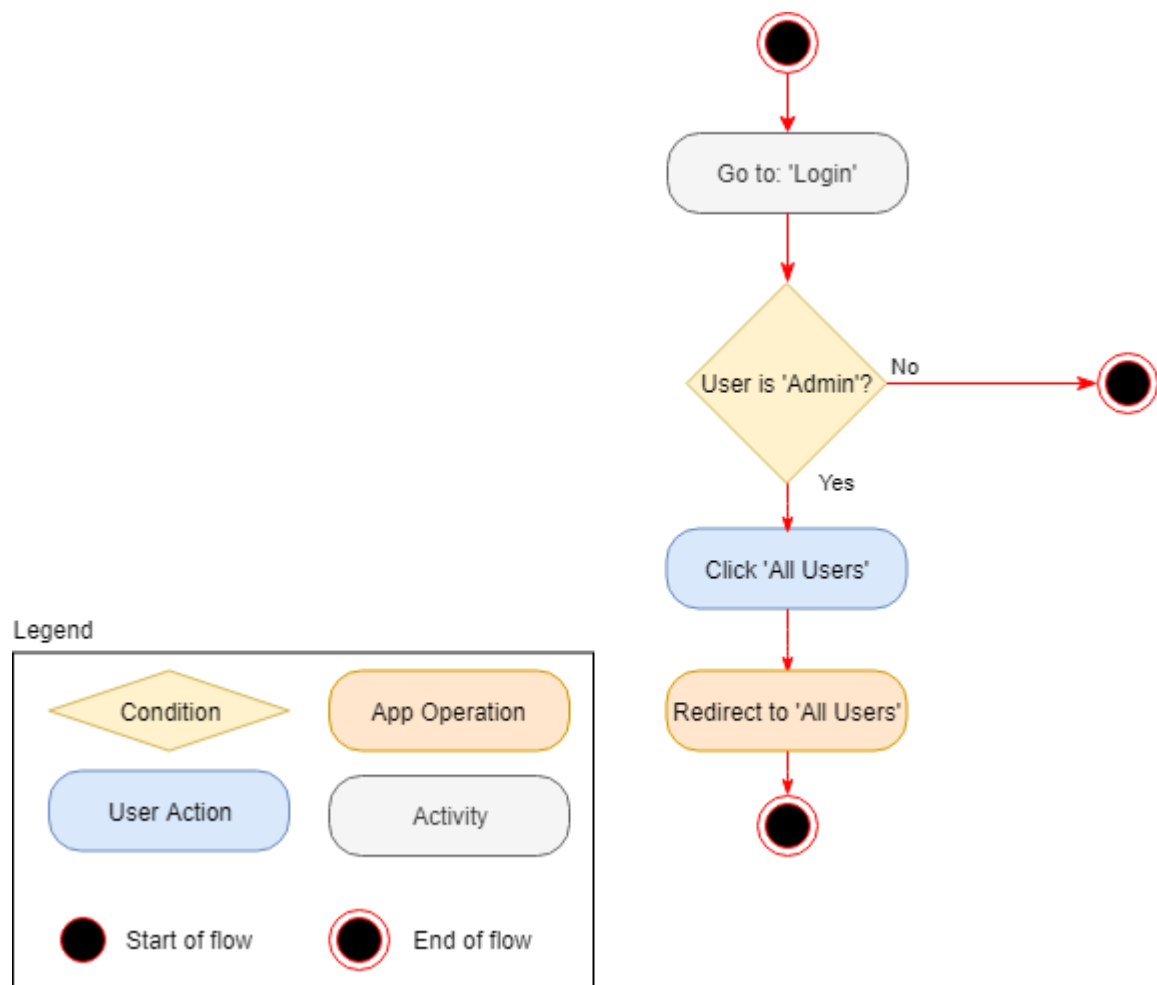


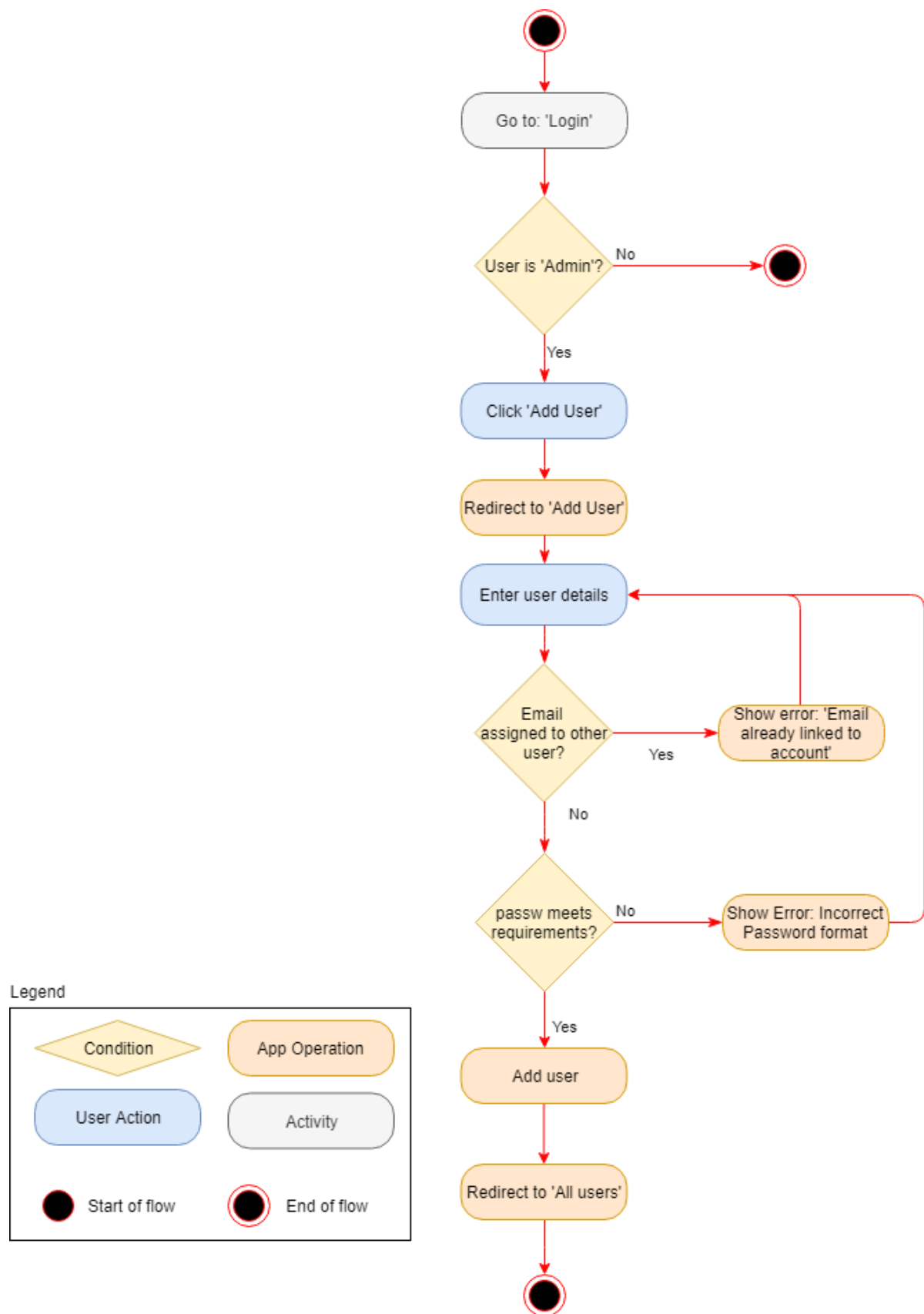


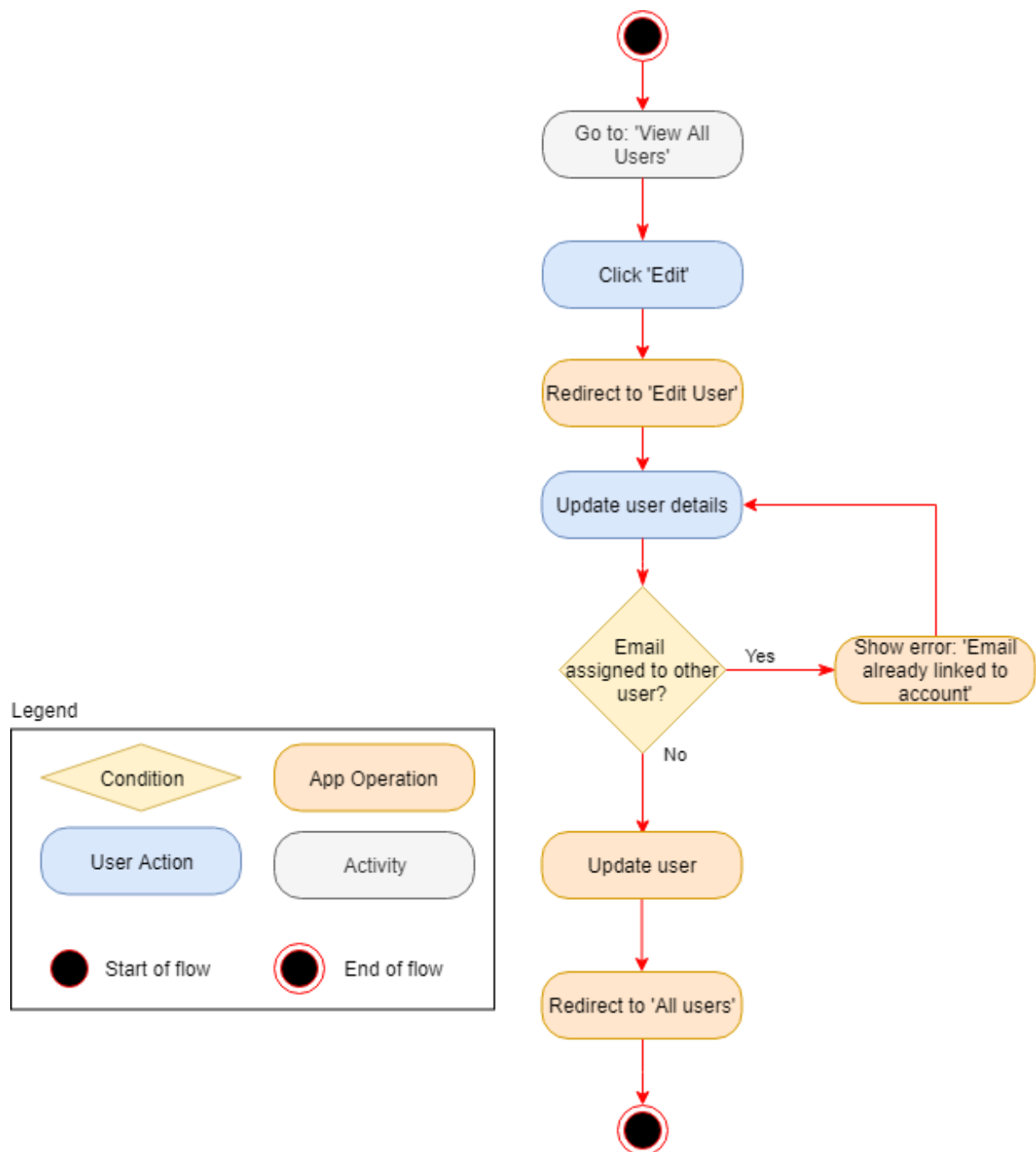


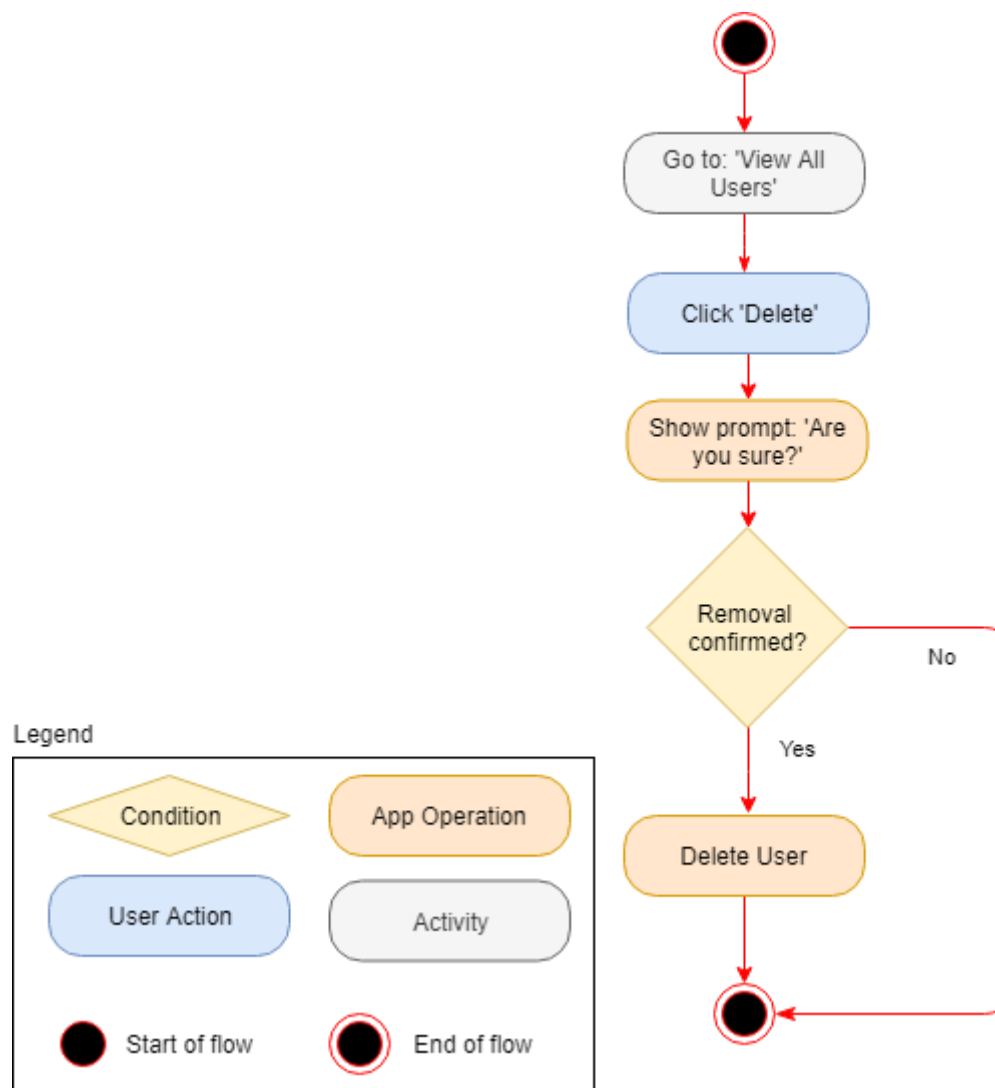










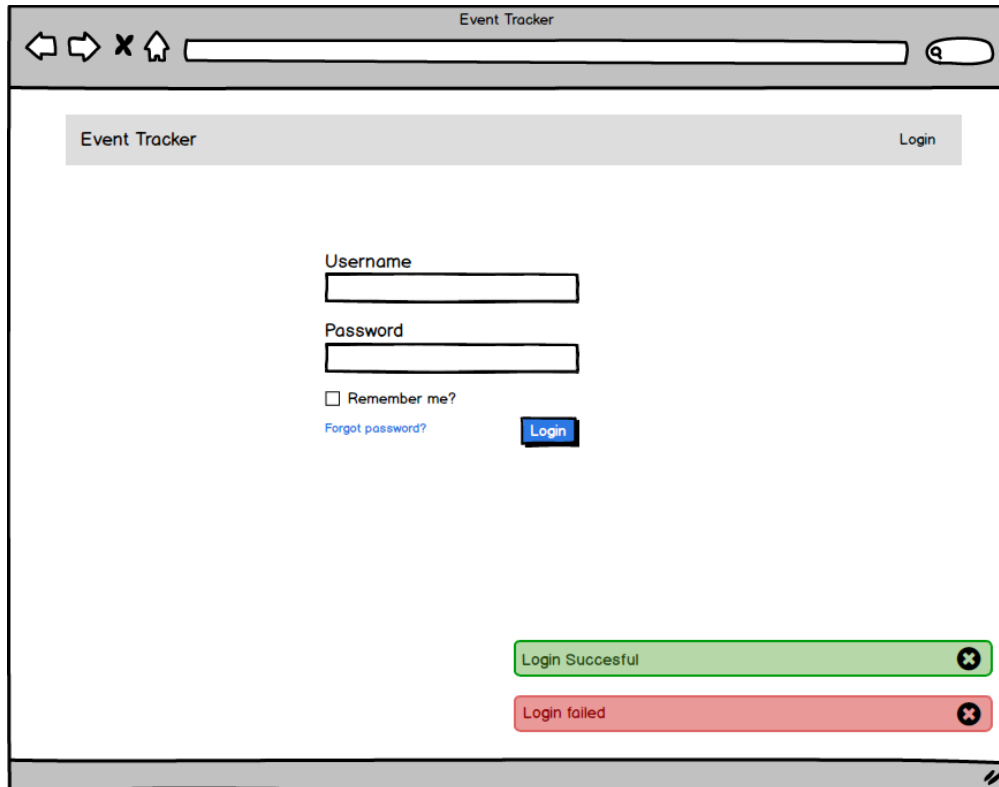


## USER INTERFACE

### USER INTERFACE MOCKUPS

#### LOGIN

Het login scherm is beschikbaar voor alle gebruikers.



A desktop browser window mockup for the 'Event Tracker' application. The browser's address bar shows 'Event Tracker' and a search icon. The page header contains 'Event Tracker' on the left and 'Login' on the right. The main content area features a login form with the following elements: a 'Username' label above a text input field, a 'Password' label above a text input field, a checkbox labeled 'Remember me?', a blue link for 'Forgot password?', and a blue 'Login' button. At the bottom right of the form area, there are two stacked notification boxes: a green one labeled 'Login Successful' and a red one labeled 'Login failed', each with a close icon (an 'x' in a circle).



A mobile phone mockup displaying the 'Event Tracker' login screen. The status bar at the top shows 'ABC' and '01:12 PM'. The app's header bar includes a hamburger menu icon and the text 'Event Tracker'. The login form is centered and contains: a 'Username' label above a text input field, a 'Password' label above a text input field, a checkbox labeled 'Remember me', a blue link for 'Forgot password?', and a blue 'Login' button. At the bottom, there are two stacked notification boxes: a green one labeled 'Login Successful' and a red one labeled 'Login failed', each with a close icon (an 'x' in a circle).

Het 'Forgot password' scherm is beschikbaar voor alle gebruikers.

The image shows a web browser window with the title 'Event Tracker'. The address bar is empty. The page header contains 'Event Tracker' on the left and a 'Login' link on the right. The main content area has the heading 'Please enter your email' followed by the text 'A link to reset your password will be sent to you'. Below this is a text input field and a blue 'Submit' button. At the bottom, there are two feedback messages: a green one that says 'Email sent' and a red one that says 'Email not found', each with a close button (an 'x' in a circle).

The image shows a mobile app interface on a smartphone. The status bar at the top shows 'ABC' and '01:17 PM'. The app header has a hamburger menu icon and the text 'Event Tracker'. The main content area displays 'Please enter your mail' and 'A link to reset your password will be sent to you'. Below this is a text input field labeled 'Email' and a blue 'Submit' button. At the bottom, there are two feedback messages: a green one that says 'Email sent' and a red one that says 'Email not found', each with a close button (an 'x' in a circle).

## RESET PASSWORD

Het reset password scherm is toegankelijk voor alle gebruikers. Dit scherm kan enkel bereikt worden via een door de applicatie gegenereerde URL die per mail naar de user wordt verstuurd nadat die een geldige email heeft ingevoerd in het 'Forgot Password' scherm.

The desktop view of the 'Event Tracker' password reset screen features a browser window with navigation icons and a search bar. The page header includes the 'Event Tracker' logo and a 'Login' link. The main content area contains two text input fields labeled 'New Password' and 'Confirm New', followed by a blue 'Reset password' button. At the bottom, there are two notification boxes: a green one stating 'Password reset' and a red one stating 'New password does not meet requirements', both with close icons.

The mobile view of the 'Event Tracker' password reset screen is displayed on a smartphone. The status bar at the top shows 'ABC', '06:51 PM', and a battery icon. The app header includes a menu icon and the 'Event Tracker' title. The layout is identical to the desktop version, with 'New Password' and 'Confirm New Password' input fields, a 'Reset Password' button, and two notification boxes at the bottom: a green 'Password reset' box and a red 'New password does not meet requirements' box.



## CHANGE PASSWORD

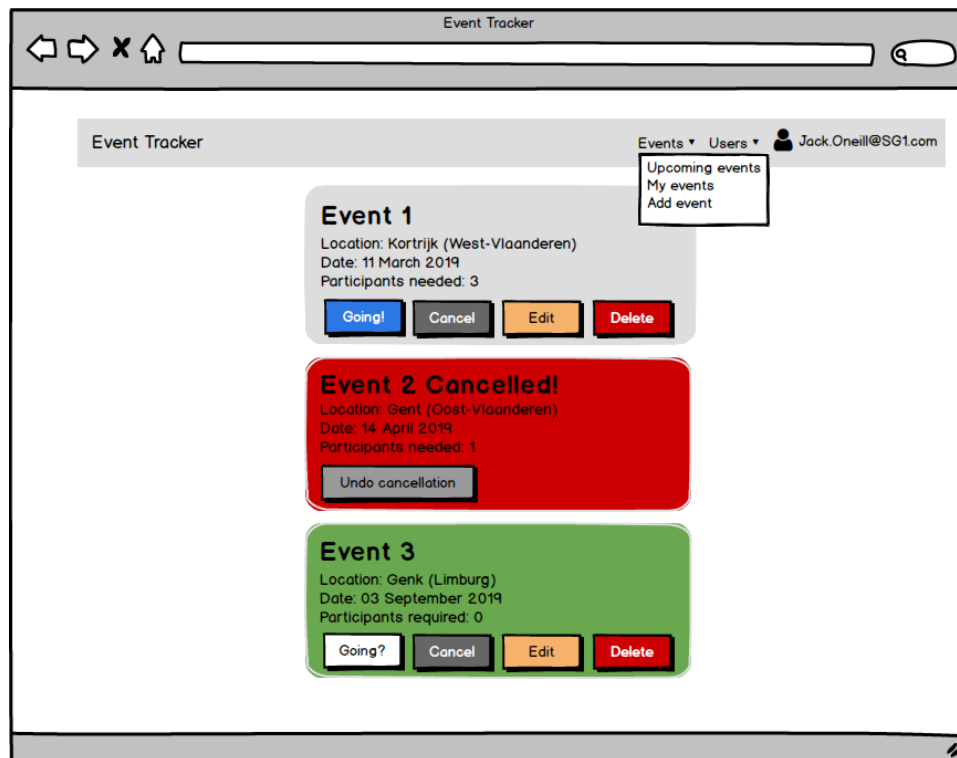
- Het 'Change password' scherm is beschikbaar voor alle gebruikers.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.

The desktop view of the 'Event Tracker' application shows a web browser window. The browser's address bar contains the text 'Event Tracker'. The application's header includes the title 'Event Tracker' on the left and navigation links 'Events', 'Users', and a user profile 'Jack.Oneill@SG1.com' on the right. A dropdown menu is open under the user profile, showing 'Change Password' and 'Logout'. The main content area contains three input fields labeled 'Current Password', 'New Password', and 'Confirm New'. Below these fields is a blue button labeled 'Update Password'. At the bottom of the page, there are three notification messages: a green one saying 'Password changed', and two red ones saying 'Old password incorrect' and 'New password does not meet requirements'. Each notification has a close button (an 'x' in a circle).

The mobile view of the 'Event Tracker' application is shown on a smartphone. The status bar at the top displays 'ABC', '01:24 PM', and a battery icon. The app's header features a hamburger menu icon, the title 'Event Tracker', and a list of navigation items: 'Events', 'Users', 'Jack.Oneill@SG1.com', 'Change Password', and 'Logout'. The main form area has three input fields labeled 'Old Password', 'New Password', and 'Confirm New Password'. A blue button labeled 'Change' is positioned below the fields. At the bottom, there are two notification messages: a green one saying 'Password Changed' and a red one saying 'Password not changed'. Both notifications include a close button (an 'x' in a circle).

## UPCOMING EVENTS

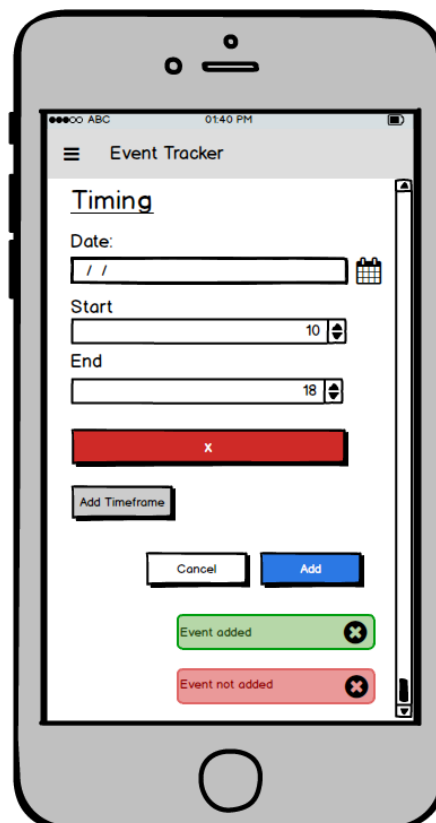
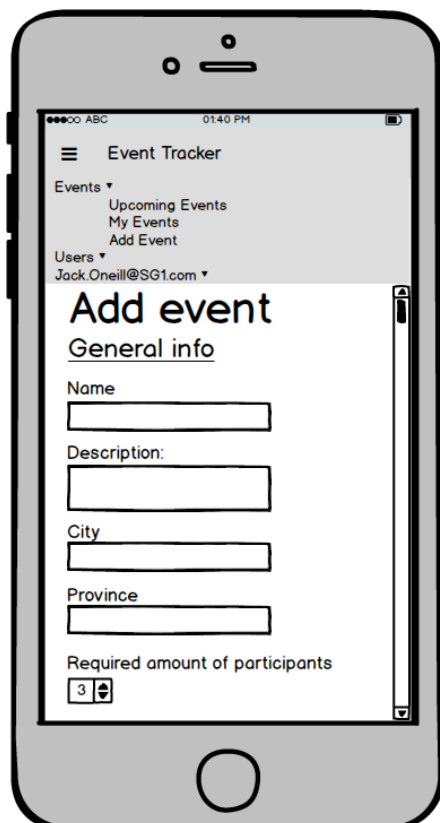
- Het 'Upcoming Events' scherm is beschikbaar voor alle gebruikers.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.
- Deze lay-out wordt gehanteerd bij zowel 'Upcoming Events' als bij 'My Events'. Enkel de buttons bij een evenement zullen in het 'My Event' scherm niet verschijnen.
- De link 'Add event' en de buttons 'Cancel', 'Edit', 'Delete' & 'Undo cancellation' zijn enkel zichtbaar voor superusers & administrators



## ADD EVENT

- Het 'Add Event' scherm is enkel beschikbaar voor superusers en administrators.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.

The desktop view of the 'Add new event' form is displayed within a browser window titled 'Event Tracker'. The form is divided into two main sections: 'General info' and 'Timing'. In the 'General info' section, there are input fields for 'Name event:', 'Description:', 'City', and 'Province', along with a 'Number of required participants' field set to 3. The 'Timing' section includes a 'Date:' field with a calendar icon, and 'Start' and 'End' time fields set to 10 and 18 respectively, with a red 'X' icon next to the end time. Below these fields is an 'Add Timeframe' button. On the right side of the form, there are two status messages: 'Event added' in a green box and 'Event not added' in a red box, both with a close icon. At the bottom right, there are 'Cancel' and 'Add' buttons. The top right of the browser window shows a user profile for 'Jack.Oneill@SG1.com'.



## EDIT EVENT

- Het 'Add Event' scherm is enkel beschikbaar voor superusers en administrators.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.

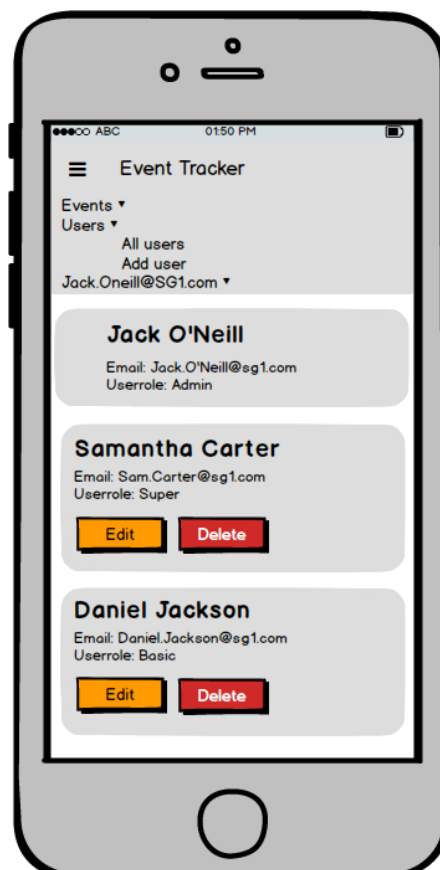
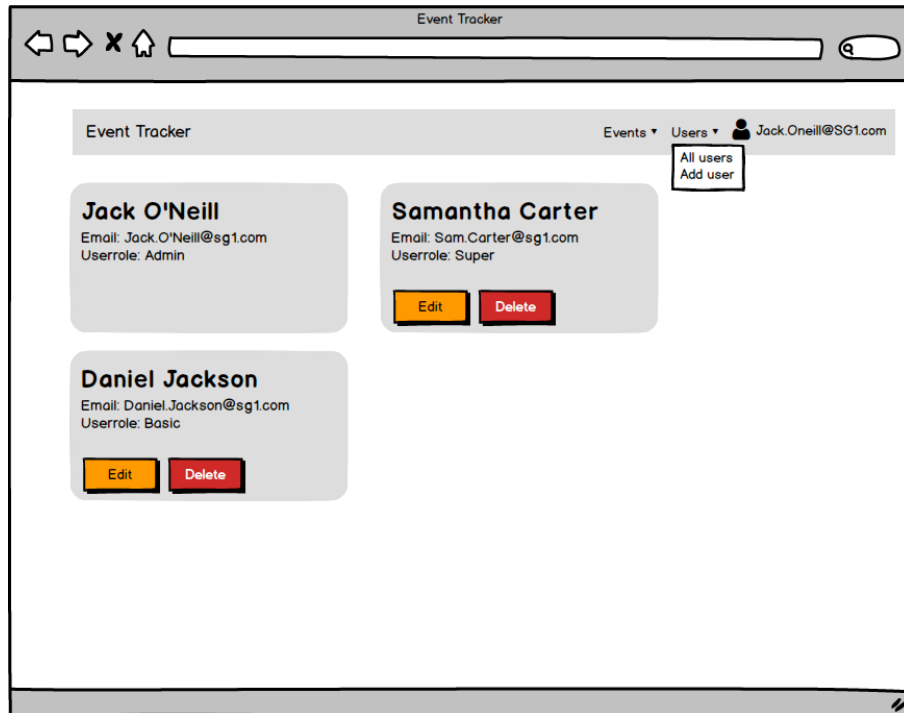
The desktop view of the 'Edit event' form is displayed within a browser window titled 'Event Tracker'. The browser's address bar is empty, and the page has a search icon in the top right corner. The application header shows 'Event Tracker' on the left and navigation links for 'Events', 'Users', and a user profile 'Jack Oneill@SG1.com' on the right. The main heading is 'Edit event'. Below it, the 'General info' section contains input fields for 'Name event:' (with placeholder 'Some Event Name'), 'Description:' (with placeholder 'Some Event Description'), 'City' (with placeholder 'Some City'), and 'Province' (with placeholder 'Some Province'). A 'Number of required participants:' is set to '3' with a spinner. The 'Timing' section includes a 'Date:' field with '03/09/1985' and a calendar icon, and 'Start' and 'End' time fields set to '10' and '18' respectively, each with a spinner and a red 'X' icon. An 'Add Timeframe' button is below. The 'Participants' section lists 'Daniel Jackson' and 'Samantha Carter', each with a red 'X' icon. On the right side of the form, there are two status messages: a green 'Event Updated' and a red 'Event not updated', both with close icons. At the bottom right are 'Cancel' and 'Update' buttons.

The mobile view of the 'Edit event' form is shown on a smartphone. The status bar at the top shows 'ABC' and '01:43 PM'. The app's navigation menu is visible on the left, showing 'Event Tracker', 'Events' (with sub-items 'Upcoming Events', 'My Events', 'Add Event'), and 'Users' (with 'Jack Oneill@SG1.com'). The 'Edit event' heading is prominent. The 'General info' section has input fields for 'Name', 'Description', 'City', and 'Province', followed by a 'Required amount of participants' spinner set to '3'. A vertical scrollbar is on the right side of the form.

This mobile view shows the 'Timing' and 'Participants' sections of the 'Edit event' form. The 'Timing' section has 'Date:' (placeholder ' / /'), 'Start' (placeholder '10'), and 'End' (placeholder '18') fields, each with a spinner and a red 'X' icon. An 'Add Timeframe' button is below. The 'Participants' section lists 'Daniel Jackson' and 'Samantha Carter', each with a red 'X' icon. At the bottom are 'Cancel' and 'Add' buttons. A vertical scrollbar is on the right side of the form.

## USERS OVERVIEW

- Het 'Users Overview' scherm is enkel beschikbaar voor administrators.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.
- Een administrator kan zijn eigen profiel niet verwijderen of aanpassen. De nodige buttons hiervoor verschijnen immers niet.



## ADD USER

- Het 'Add user' scherm is enkel beschikbaar voor administrators.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.

The desktop view of the 'Event Tracker' application shows a form for adding a new user. The form is titled 'Event Tracker' and includes a navigation bar with 'Events' and 'Users' dropdown menus, and a user profile icon for 'Jack.Oneill@SG1.com'. The form fields are: 'First name', 'Last name', 'Email', 'Password', 'Confirm Password', and 'Userrole' (a dropdown menu currently set to 'Basic'). To the right of the form, there are two status messages: 'User added' (green) and 'User not added' (red), both with close buttons. At the bottom right, there are 'Cancel' and 'Add' buttons.

The mobile view of the 'Event Tracker' application shows the same 'Add User' form on a smartphone screen. The form is titled 'Event Tracker' and includes a navigation bar with a hamburger menu icon and the text 'Event Tracker'. The form fields are: 'First name', 'Last name', 'Email', 'Password', 'Confirm Password', and 'Userrole' (a dropdown menu currently set to 'Basic'). At the bottom, there are 'Cancel' and 'Add' buttons.

## EDIT USER

- Het 'Edit user' scherm is enkel beschikbaar voor administrators.
- Het dropdown menu 'Users' is enkel zichtbaar voor administrators.

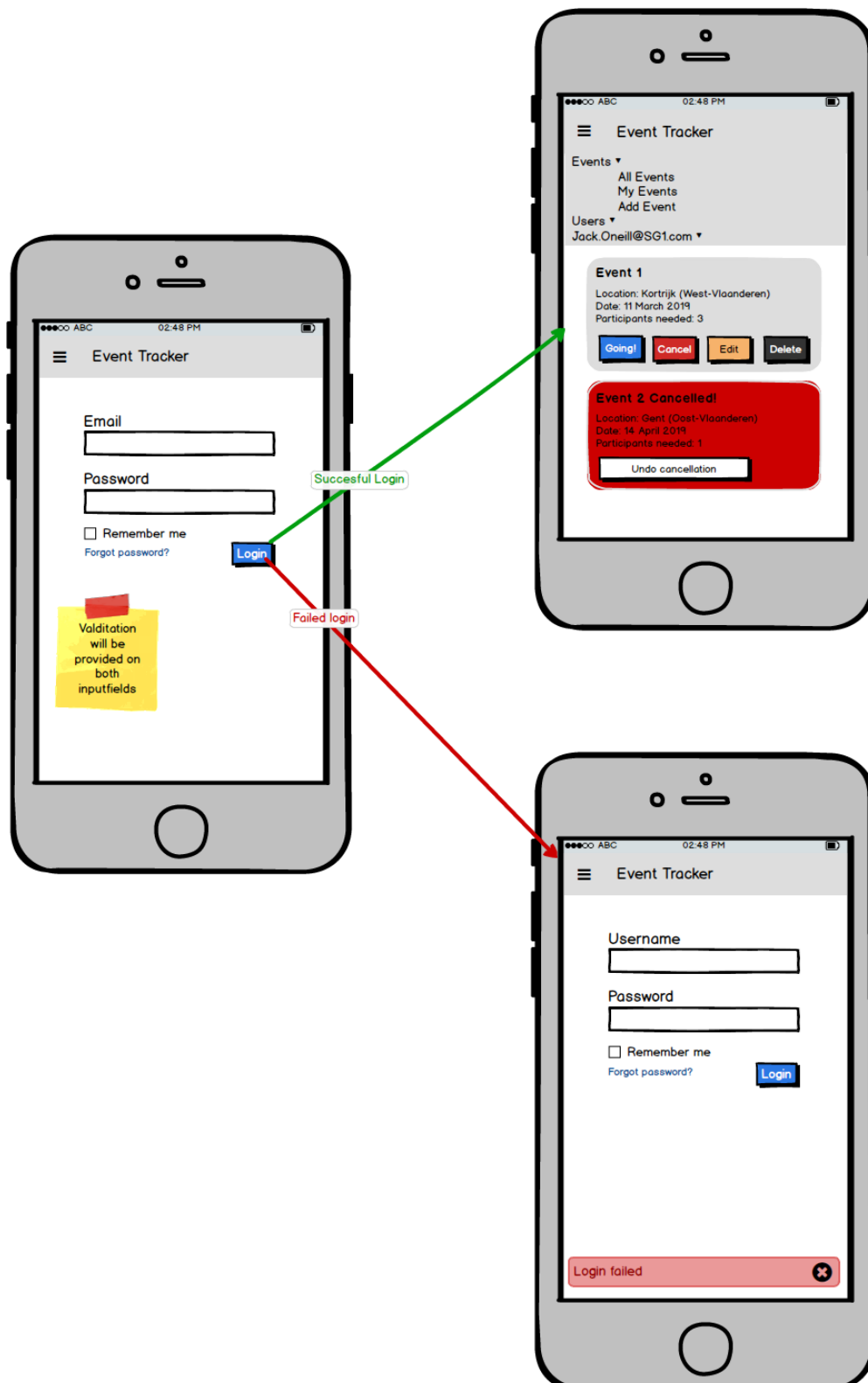
The desktop view of the 'Event Tracker' application shows a web browser window. The title bar reads 'Event Tracker'. The browser's address bar is empty. The page header includes 'Event Tracker' on the left, and 'Events ▾ Users ▾ Jack.Oneill@SG1.com ▾' on the right. The main form contains four input fields: 'First name' with the value 'Daniel', 'Last name' with the value 'Jackson', 'Email' with the value 'Daniel.Jackson@sg1.com', and 'Userrole' with a dropdown menu showing 'Basic'. At the bottom right of the form, there are two status messages: a green one that says 'User updated' and a red one that says 'User not updated', both with a close button (✕). Below these messages are two buttons: 'Cancel' and 'Update'.

The mobile view of the 'Event Tracker' application is shown on a smartphone. The status bar at the top displays 'ABC' and '02:46 PM'. The app's header shows a hamburger menu icon and the text 'Event Tracker'. The form fields are identical to the desktop version: 'First name' (Daniel), 'Last name' (Jackson), 'Email' (Daniel.Jackson@sg1.com), and 'Userrole' (Basic). At the bottom, there are 'Cancel' and 'Add' buttons. The 'Add' button is highlighted in blue.

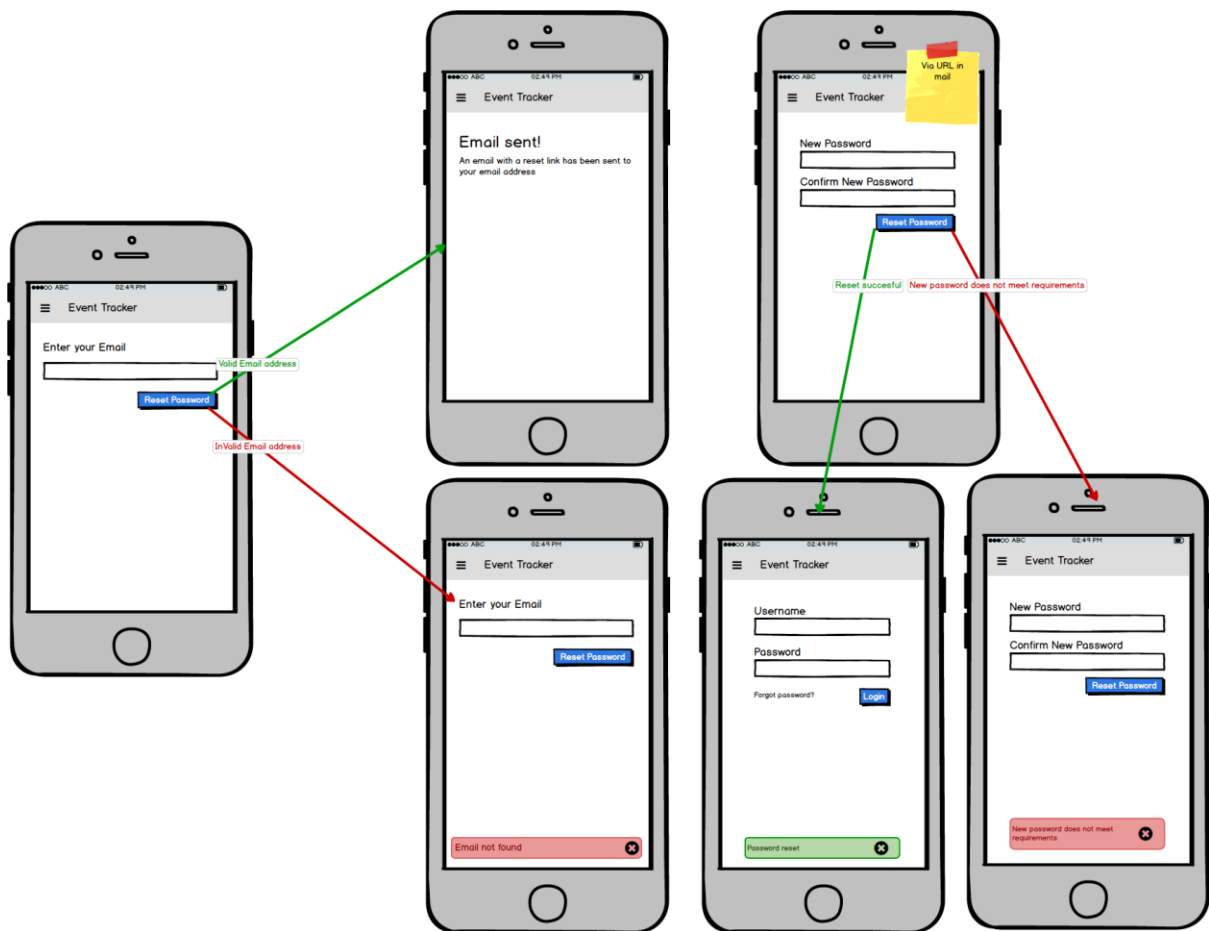
## STORYBOARDS

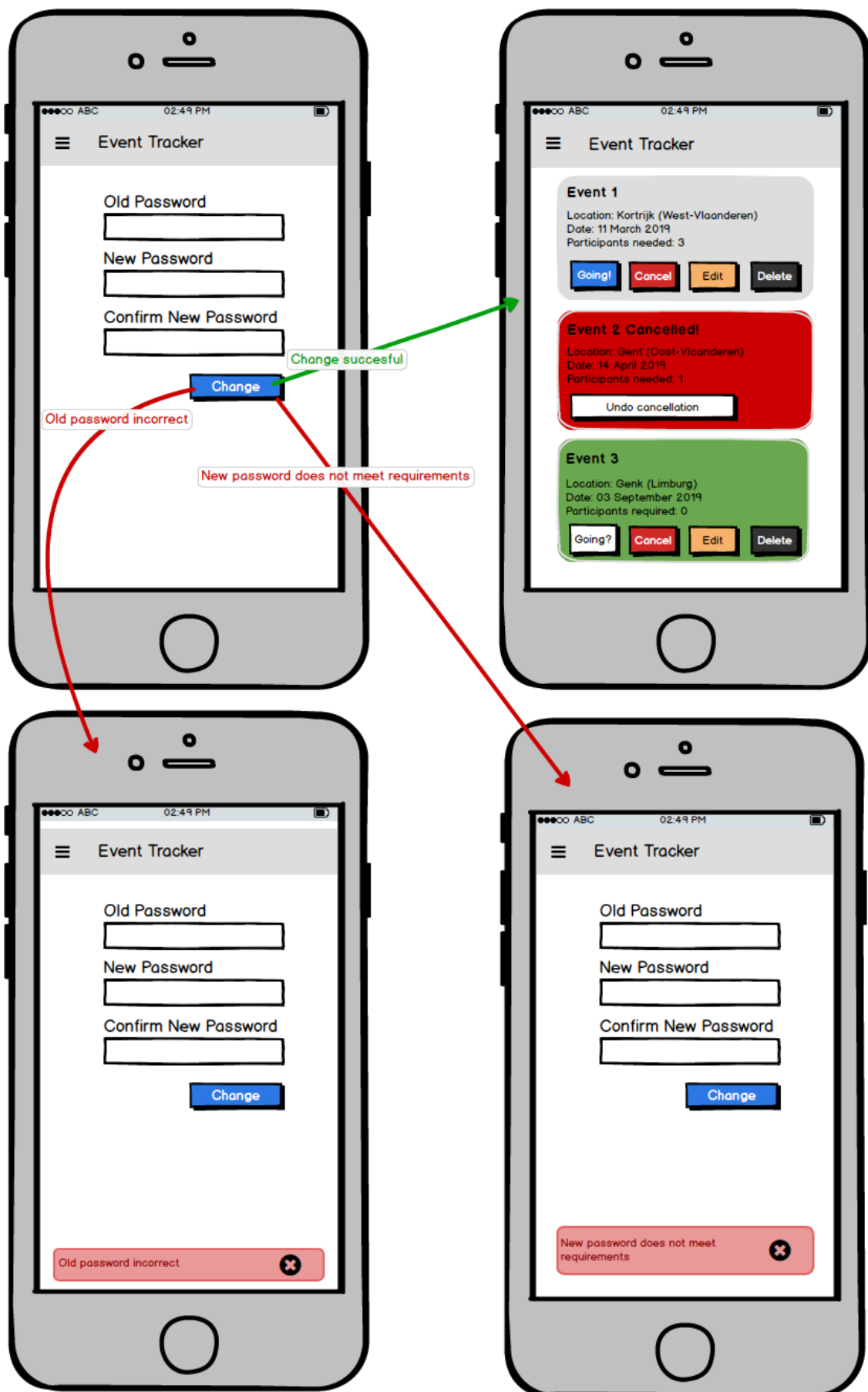
Op vlak van navigatie is er geen wezenlijk verschil tussen de mobiele UIs en desktop UIs. Om het simpel te houden, wordt bijgevolg enkel de mobiele UI gebruikt bij onderstaande storyboards. Tevens bevatten alle UIs in onderstaande storyboards alle functionaliteiten die de applicatie aanbiedt. Sommige controls of UIs zijn dus niet toegankelijk voor alle gebruikers. Dit is waar nodig gespecificeerd.

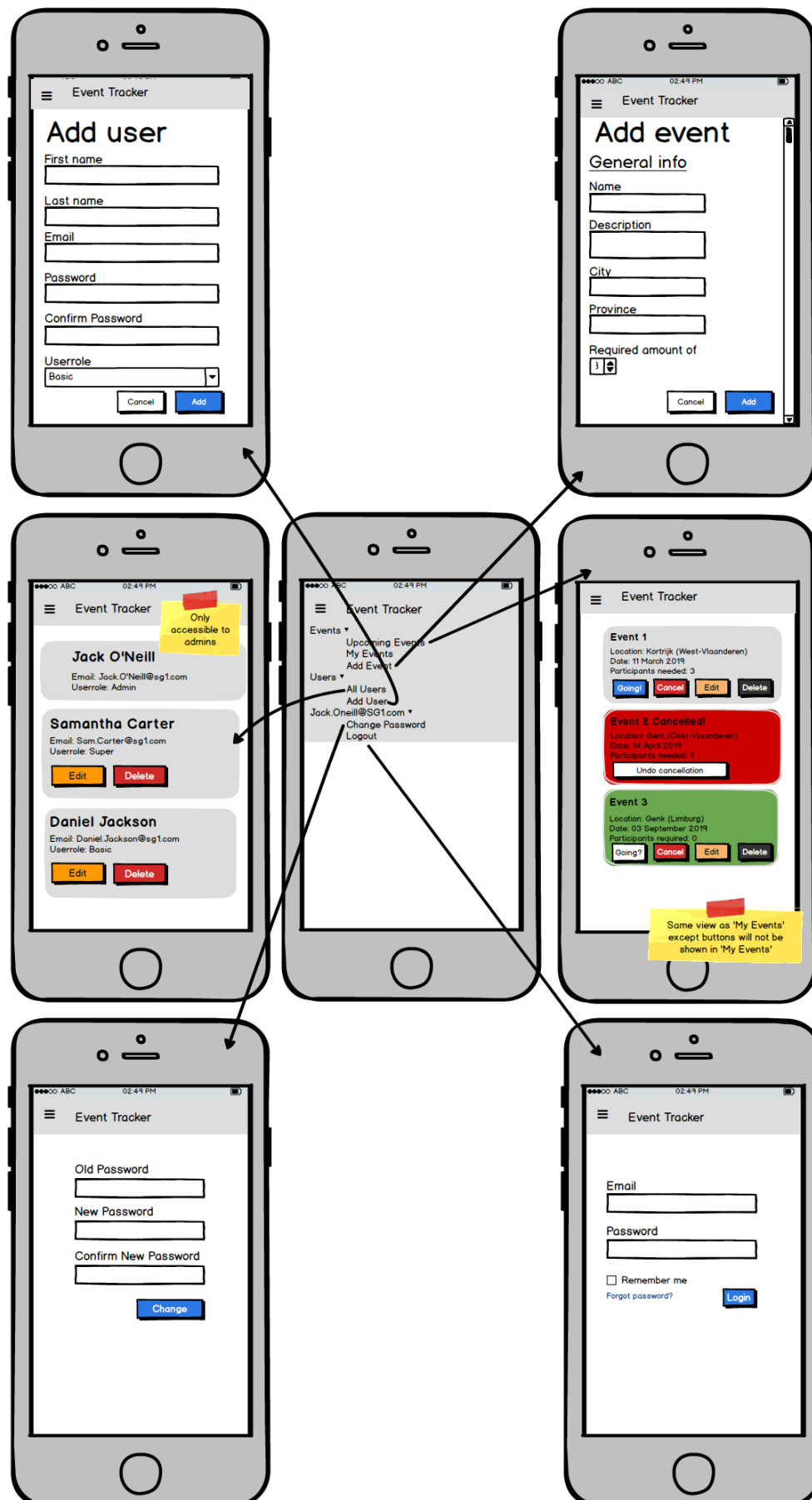
### LOGIN



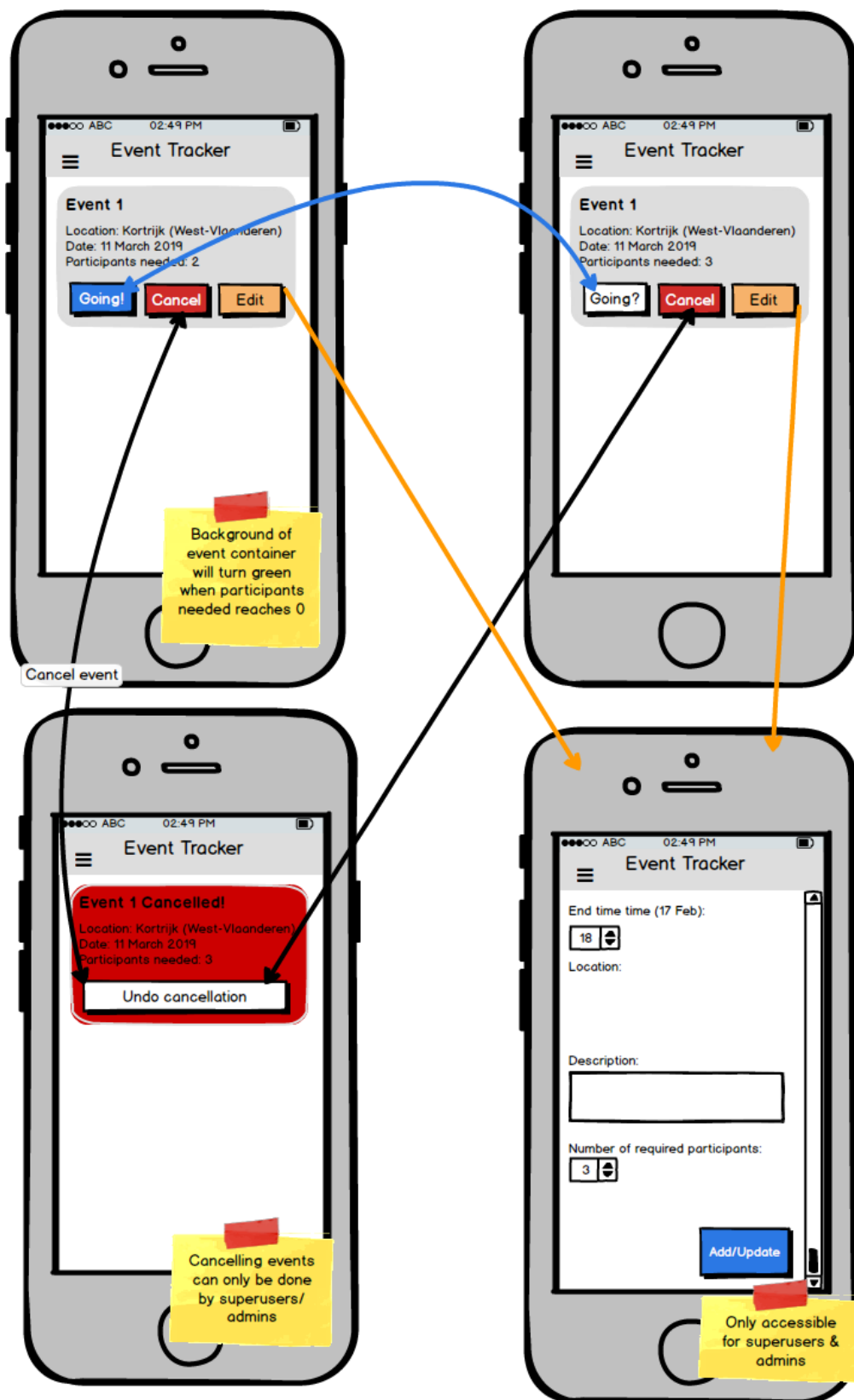


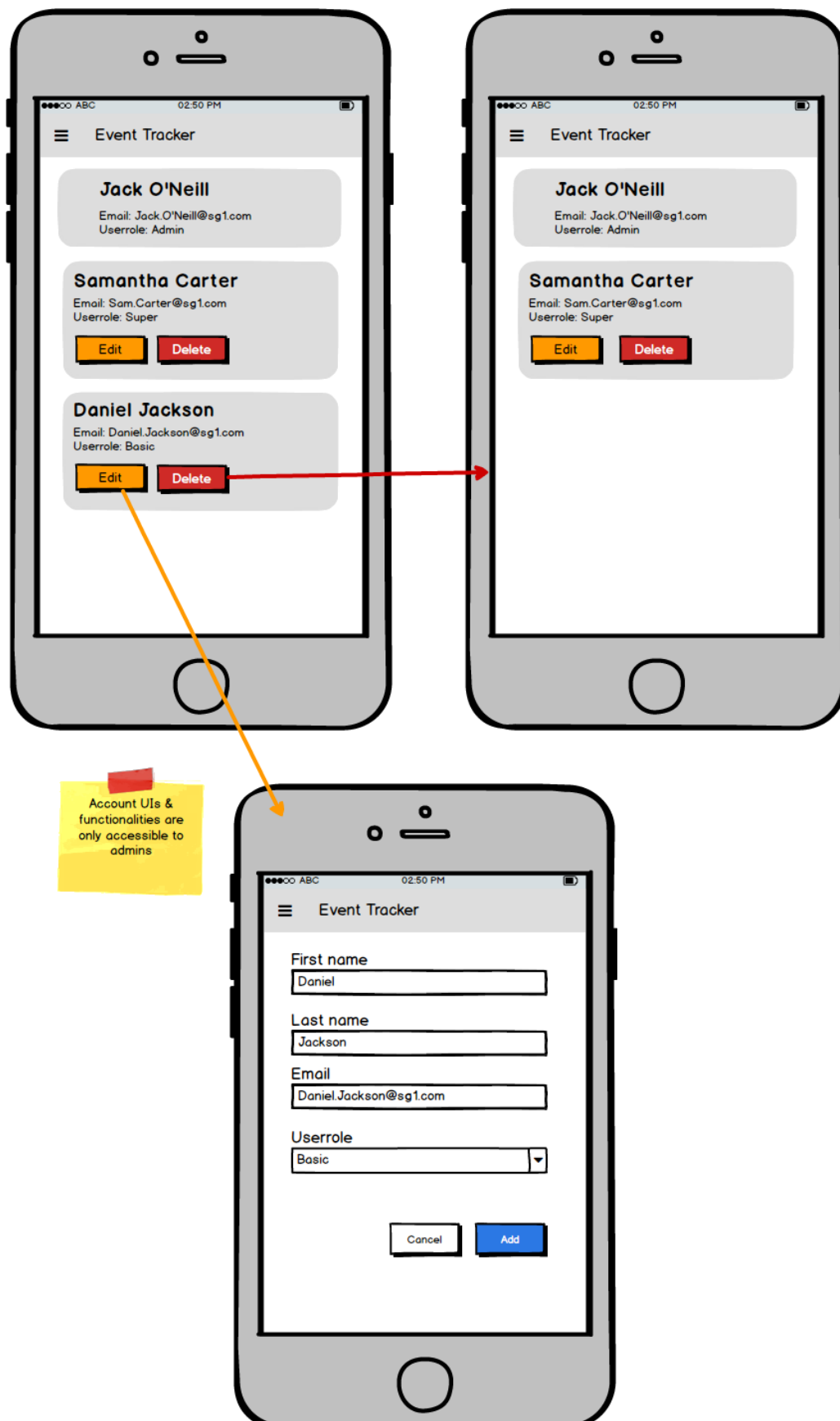






Gezien onderstaand storyboard reeds veel interacties kent, is de 'Delete event' functionaliteit hier niet opgenomen. De gebruiker wordt na het verwijderen van een evenement simpelweg teruggestuurd naar een opgedateerd overzicht van de nog resterende evenementen.





## ONTWIKKELINGSTACK

### DOELPLATFORM

---

- OS: Cross Platform
- Device: All devices
- Doel is een zo breed mogelijk publiek aan te spreken met een zo beperkt mogelijke code base
- Focus op mobile-friendly user experience door gebruik van Bootstrap 4 (Mobile First)

### TECHNOLOGIEËN

---

#### FRONT END

---

- [HTML 5](#)
- [CSS 3](#)
- [SaSS](#)
- [JavaScript](#)
- [Jquery](#) - v3.4.0
- [Jquery Validation](#) - v1.19.0
- [Jquery Validation Unobtrusive](#) - v3.2.11
- [Popper.js](#) - v2.0.0-next.4
- [Bootstrap](#) - v4.3.1
- [Font Awesome](#) - v4.7.0

#### BACK END

---

- [ASP.NET Core MVC](#) - v2.1
- [ASP.NET Core Identity](#) - v2.1.1
- [Entity Framework Core](#) - v2.1.8
- [MailKit](#) - v2.1.5.1
- [BeginCollectionItemCore](#) – v1.0.8

### TOOLS

---

- IDE: [MS Visual Studio 2019 Community Edition](#)
- UI Mockups & Storyboards: [Balsamiq Mockups 3](#)
- Diagrams: [Draw.io](#)
- Visual Studio Tooling: [Bundler & Minifier](#) ([Mads Kristensen](#)) – Version 2.9.406

## ARCHITECTUUR

### APPLICATION LAYERS

---

Event Tracker bestaat uit 4 verschillende projecten. Elk project heeft zijn eigen verantwoordelijkheid.

De 4 projecten zijn

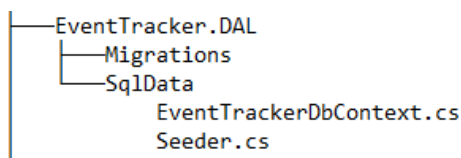
- de Data Access Layer (EventTracker.DAL)
- de Models Layer (EventTracker.Models)
- de Services Layer (EventTracker.Services)
- de Web Application Layer (EventTracker.Web)

### DATA ACCESS LAYER

---

De Data Access Layer is verantwoordelijk voor het persisteren van de verschillende entities naar de onderliggende database. Hierbij wordt gebruik gemaakt van Entity Framework Core.

Het project bestaat uit volgende files & folders:

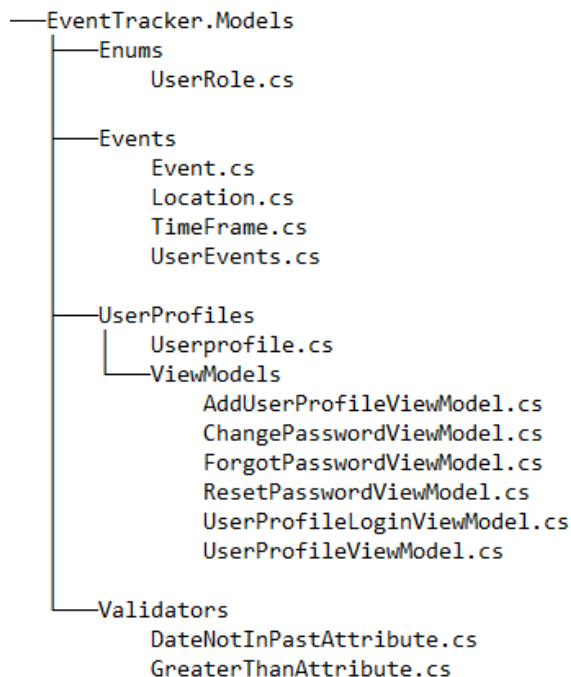


De Migrations folder wordt gebruikt door Entity Framework en zorgt ervoor dat de structuur van de onderliggende database compatibel blijft met de entities in de applicatie. Telkens wanneer de structuur van deze entities wordt aangepast in de code, moet een nieuwe migration ervoor zorgen dat de database mee evolueert.

De SqlData folder bevat twee files:

- EventTrackerDbContext.cs: Hier wordt Entity Framework geconfigureerd door de verschillende entities en hun onderlinge relaties te verduidelijken. Dit wordt verder beschreven in het hoofdstuk '[Database Diagram](#)'
- Seeder.cs: Een class die verantwoordelijk is voor het populeren van de database met testdata. Deze logica wordt enkel uitgevoerd in de 'Development' omgeving en wanneer de applicatie detecteert dat bepaalde tables van de database leeg zijn.

De Models Layer is verantwoordelijk voor het beheren van zowel de EntityModels als de ViewModels. Het project bestaat uit volgende files & folders:



Er worden in essentie slechts twee hoofdentities gebruikt die op hun beurt enkele subentiteiten bevatten:

- Event:
  - Deze omvat alle details van een evenement.
  - Hierbij wordt gebruik gemaakt de subentiteiten 'Location', 'TimeFrame' en 'UserEvents'.
- UserProfile:
  - Deze omvat alle details van een gebruiker van de applicatie.
  - Deze klasse erft alle eigenschappen over van de 'IdentityUser' klasse, een klasse die gebruikt wordt door ASP.NET Core Identity, een framework dat authenticatie & autorisatie beheert.

Belangrijk om hierbij te vermelden:

- Zowel de entity 'Event' als 'UserProfile' maken gebruik van 'UserEvents'. Er bestaat namelijk een many-to-many relatie tussen deze entiteiten. (Zie ook hoofdstuk [Database Diagram](#)).
- Voor de entity 'UserProfile' bestaan er een resem aan ViewModels. Voor de entity 'Event' bestaat er geen enkel ViewModel. De reden hiertoe is, kort door de bocht uitgelegd, dat de 'Event' entity niet gevoelig is voor overposting. Dit wordt verder verduidelijkt in het hoofdstuk [Overposting](#).

Naast de twee 'hoofdentities' en hun 'subentities' zijn er nog twee 'model helpers'.

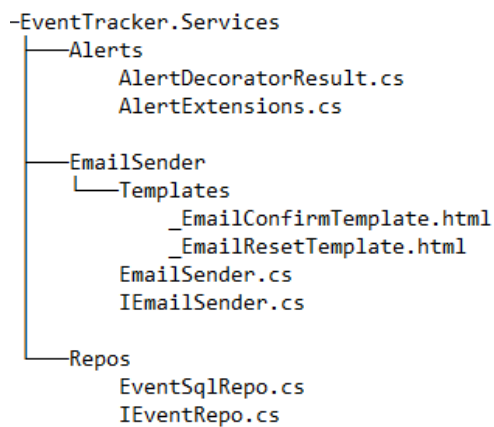
- UserRole: Een simpele enumeratie om de autorisatieniveaus (basic, super, admin) te regelen.
- Validators: Twee uitbreidingen op 'Attributes' die helpen bij het valideren van 'TimeFrames'
  - DateNotInPastAttribute.cs: Zorgt ervoor dat een nieuw aangemaakt of een aangepast evenement geen datum in het verleden kan hebben. Het houdt immers geen steek om evenementen uit het verleden aan te maken.
  - GreaterThanAttribute.cs: Zorgt ervoor dat het startuur van een 'TimeFrame' niet groter is dan het einduur



## DE SERVICES LAYER

---

De Services Layer is verantwoordelijk voor het aanleveren van enkele services. Het project bestaat uit volgende files & folders:



- De 'Alert' service is verantwoordelijk voor het beheren van alle notifications doorheen de applicatie. Het bouwt verder op het Bootstrap Alert component. Deze service wordt verder uitgelegd in het hoofdstuk [Alerts](#)
- De 'Emailsender' service is verantwoordelijk voor het versturen van zowel de 'Confirmation Email' bij het aanmaken van een nieuwe gebruiker als de 'Reset Email' bij het aanvragen van een paswoord reset.
- De 'Repos' service is verantwoordelijk voor alle CRUD functionaliteit wat betreft de entiteit 'Event'. De entiteit 'UserProfile' heeft geen nood aan dergelijke service gezien deze gebruik maakt van het Identity framework voor database persistentie.

## DE WEB APPLICATION LAYER

---

De Web Application Layer is veruit de meest omvattende layer. Het is verantwoordelijk voor onder andere:

- het aanbieden van statische content via de wwwroot folder
- het aanleveren van de UI via Controllers & Views
- beperkte business logica
- de startup logica van de applicatie
- configuratie details
- het beheren van NuGet dependencies
- het bundlen & minifyen van javascript & css files

Het project bestaat uit volgende files & folders:

- EventTracker.Web
  - appsettings.Development.json
  - appsettings.json
  - bundleconfig.json
  - compilerconfig.json
  - compilerconfig.json.defaults
  - package.json
  - Program.cs
  - Startup.cs
- Controllers
  - AccountController.cs
  - ErrorController.cs
  - EventsController.cs
  - UserProfilesController.cs
- Extensions
  - ServiceExtensions.cs
  - UrlHelperExtensions.cs
- Views
  - \_ViewImports.cshtml
  - \_ViewStart.cshtml
  - Account
    - ChangePassword.cshtml
    - ConfirmEmail.cshtml
    - ForgotPassword.cshtml
    - Login.cshtml
    - ResetPassword.cshtml
    - \_ViewStart.cshtml
  - Error
    - EventNotFound.cshtml
    - PageNotFound.cshtml
    - ServerError.cshtml
    - UserProfileNotFound.cshtml
    - \_ViewStart.cshtml
  - Events
    - AddEvent.cshtml
    - EditEvent.cshtml
    - MyEvents.cshtml
    - UpcomingEvents.cshtml
  - Partials
    - \_ActionButtons.cshtml
    - \_CancelModal.cshtml
    - \_DeleteModal.cshtml
    - \_EventAddEditDetails.cshtml
    - \_EventDetails.cshtml
    - \_Participants.cshtml
    - \_Timeframes.cshtml
  - Shared
    - \_AccountLayout.cshtml
    - \_ErrorLayout.cshtml
    - \_Layout.cshtml
  - Partials
    - \_Navbar.cshtml
    - \_StatusMessages.cshtml
  - UserProfiles
    - AddUserProfile.cshtml
    - AllUserProfiles.cshtml
    - EditUserProfile.cshtml
  - Partials
    - \_DeleteModal.cshtml
    - \_UserProfileDetails.cshtml
- wwwroot
  - css
    - site.css
    - site.min.css
    - site.scss
    - \_alerts.scss
    - \_buttons.scss
    - \_errorViews.scss
    - \_events.scss
    - \_layout.scss
    - \_normalize.scss
    - \_validation.scss
  - fonts
    - fontawesome-webfont.eot
    - fontawesome-webfont.svg
    - fontawesome-webfont.ttf
    - fontawesome-webfont.woff
    - fontawesome-webfont.woff2
    - FontAwesome.otf
  - js
    - site.js
    - site.min.js
    - vendor.min.js

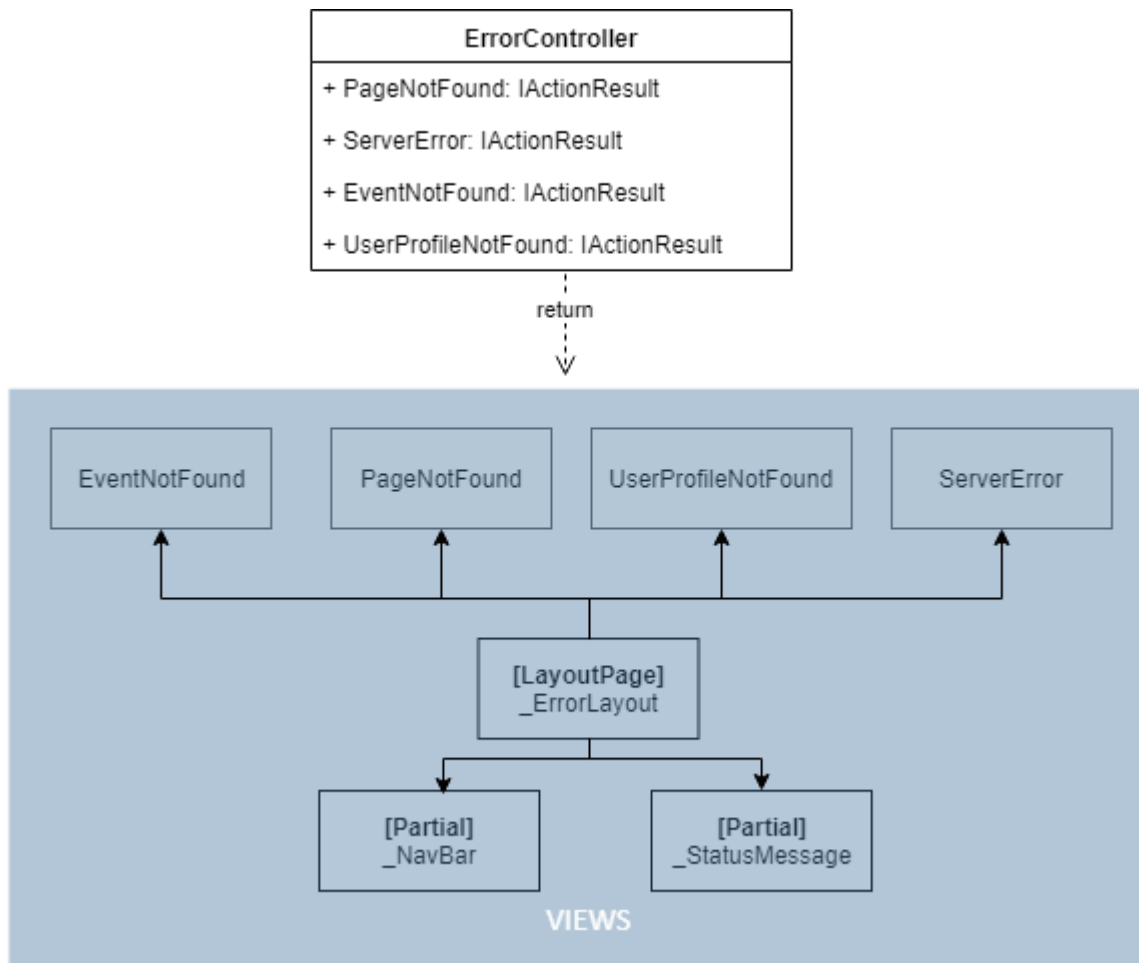
Gezien de uitgebreide collectie aan folders & files, volgt hieronder een eerder beperkte uitleg. Waar nodig wordt dieper ingegaan op bepaalde zaken of doorverwezen naar een ander hoofdstuk.

- .Json files:
  - Appsettings.json: configuratie van connectionstring & smtpclient voor productieomgeving
  - appsettings.Development.json: configuratie van connectionstring & smtpclient voor development omgeving
  - bundleconfig.json: bundles & minifies javascript files
  - compilerconfig.json: converteert scss naar css
  - package.json: basis applicatie configuratie + NuGet dependencies beheer
- Program.cs & Startup.cs:
  - Startup logica voor applicatie (toevoegen van gebruikte services + configuratie van die services)
- Controllers
  - Verantwoordelijk voor beperkte business logica + aanbieden van views
  - Controllers voor
    - Account: authenticatie & autorisatie logica
    - Error: userfriendly errorhandling
    - Events: alle logica omtrent evenementen
    - UserProfile: CRUD logica omtrent users
- Extensions
  - ServiceExtensions: Helperklasse om configuratie van bepaalde services te groeperen & te extrapoleren uit de 'Startup' klasse
  - UrlHelperExtensions: Helperklasse om account confirmatie & reset links te creëren
- Views
  - Alle UI logica
- wwwroot
  - Statische files die aan gebruiker worden aangeboden
  - css files, javascript files & fonts

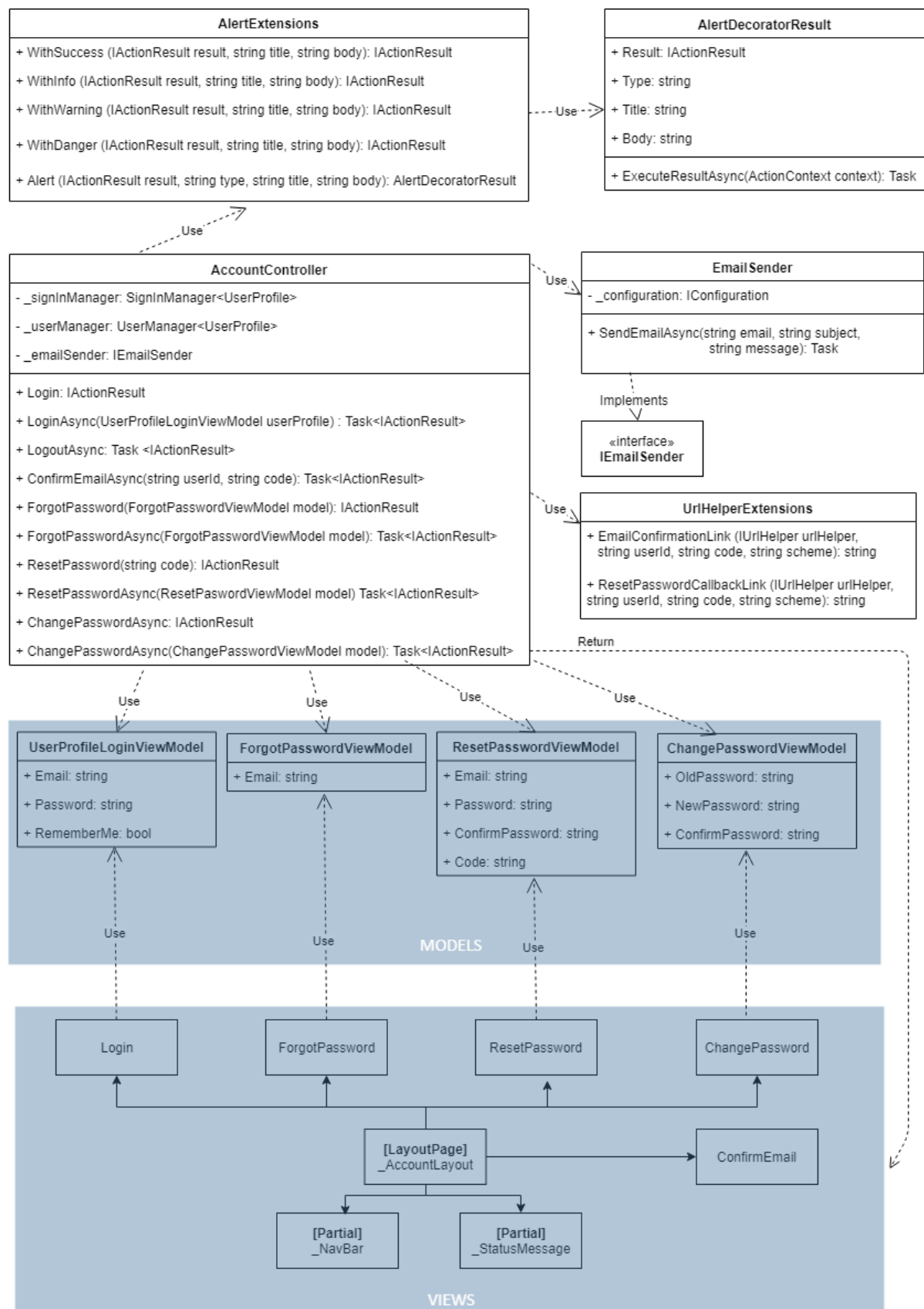
## CLASS DIAGRAM

Een klassediagram opstellen van de gehele applicatie vertaalt zich al snel in een complex & chaotisch diagram. Klassediagrammen per controller, die hieronder te vinden zijn, lijkt een betere keuze. Enkele elementen zijn echter alsnog verwijderd om de complexiteit verder te beperken. Deze elementen zijn onderdelen van frameworks zoals de UserManager klasse of SignInManager klasse van het Identity Framework of de Program & Startup klasse van ASP.NET Core. Een overzicht van de fields en methods van deze klassen kunnen in de Microsoft documentatie worden teruggevonden.

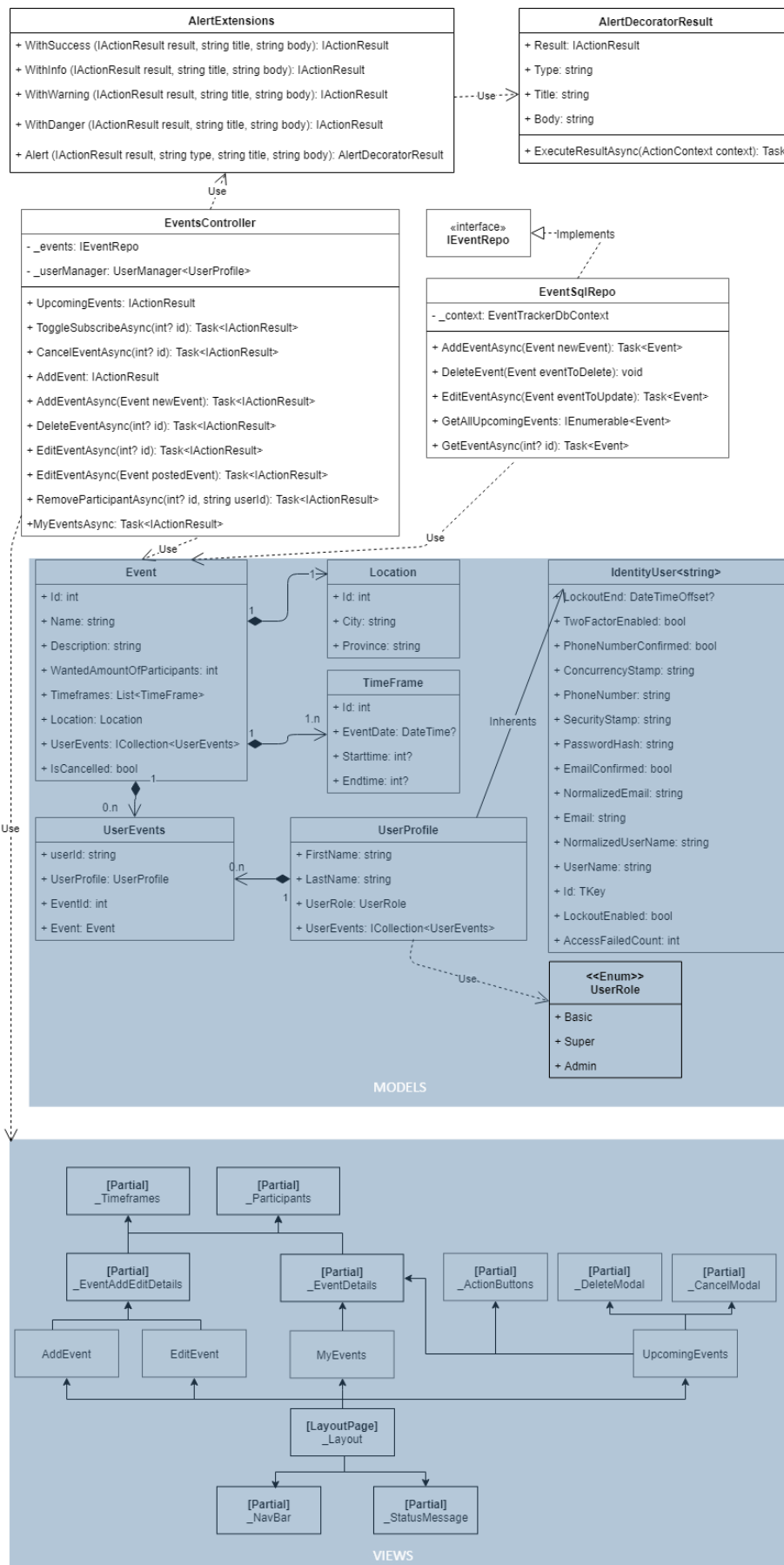
## ERRORCONTROLLER



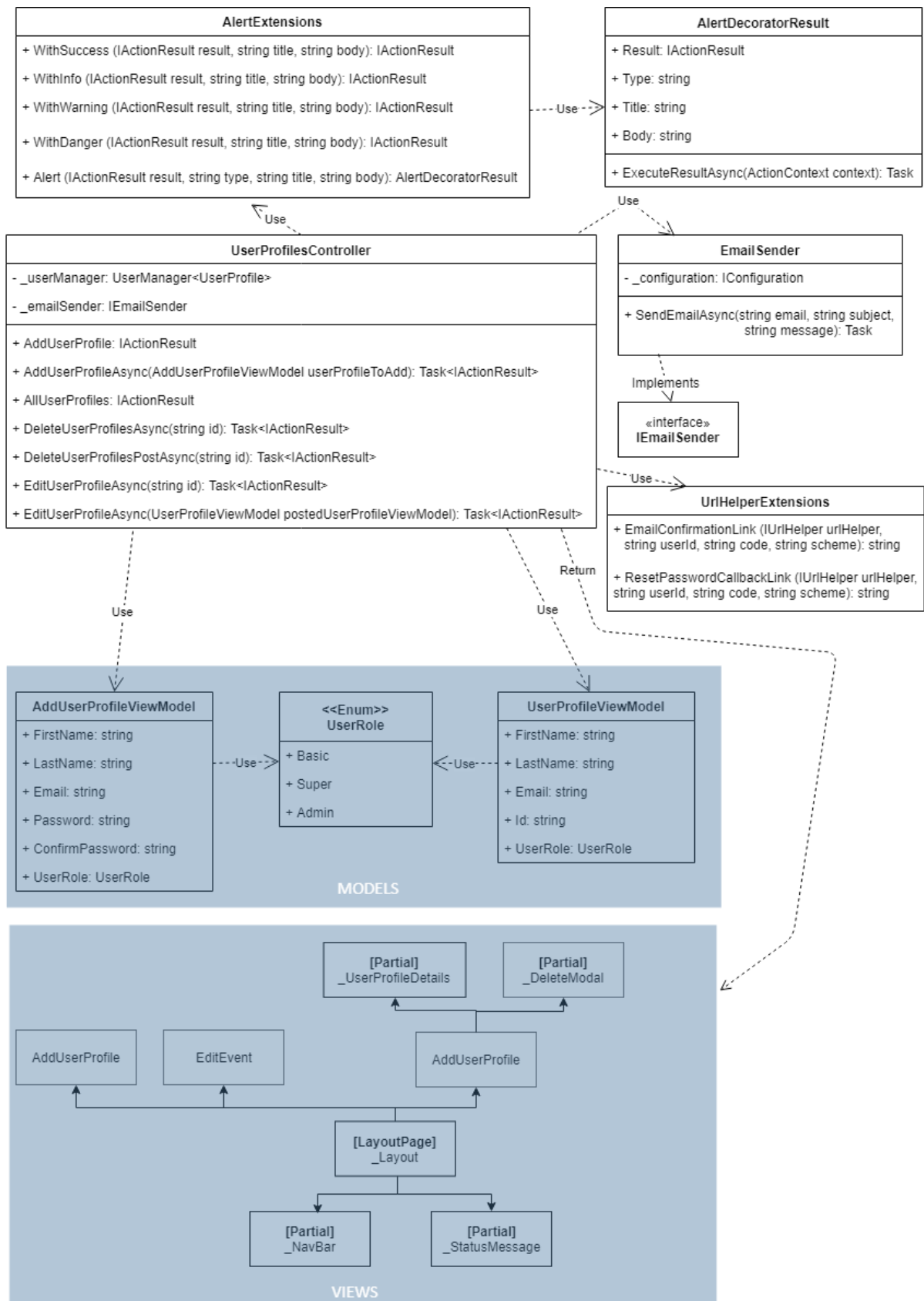
## ACCOUNTCONTROLLER



## EVENTSCONTROLLER

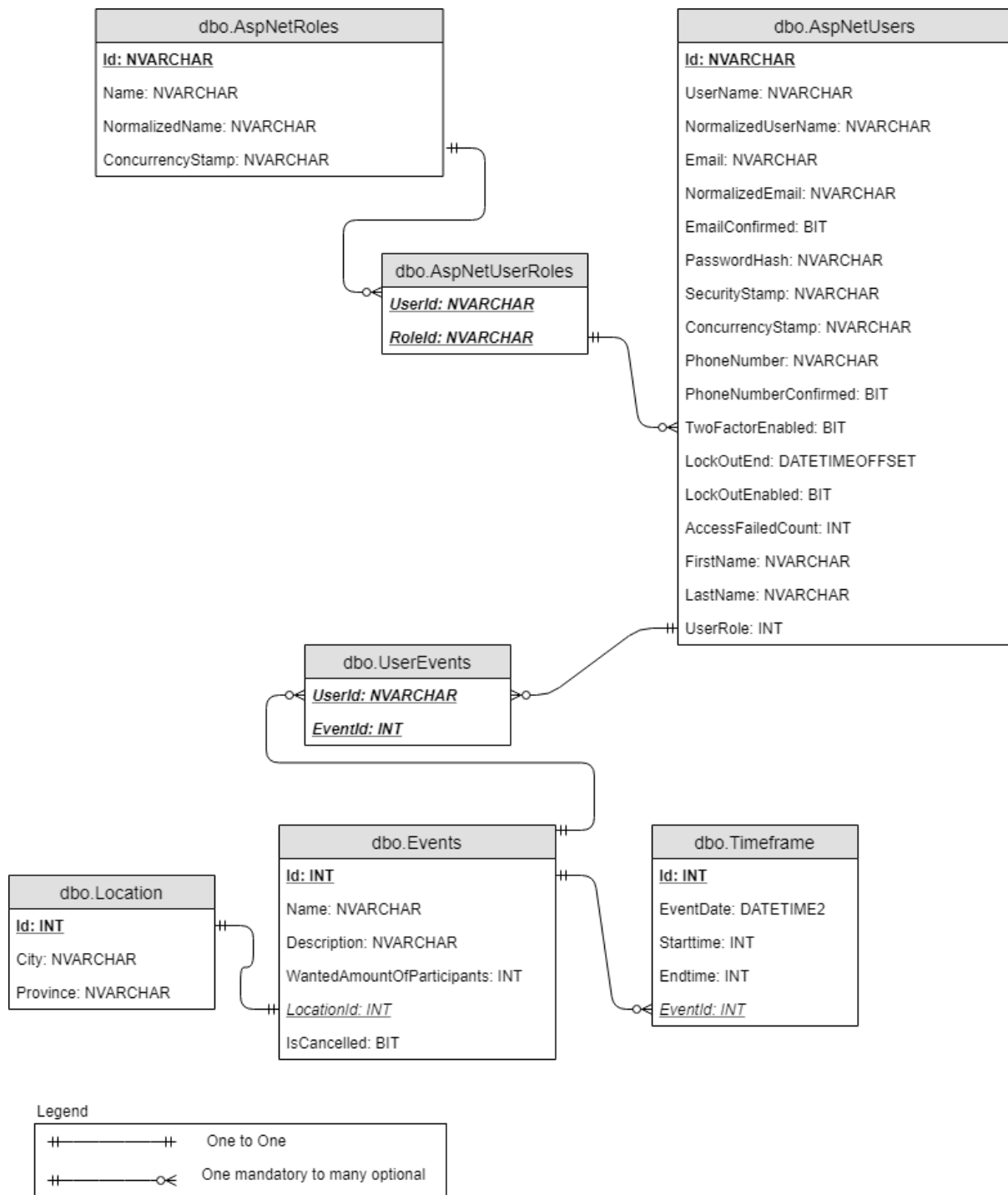


## USERPROFILESCONTROLLER



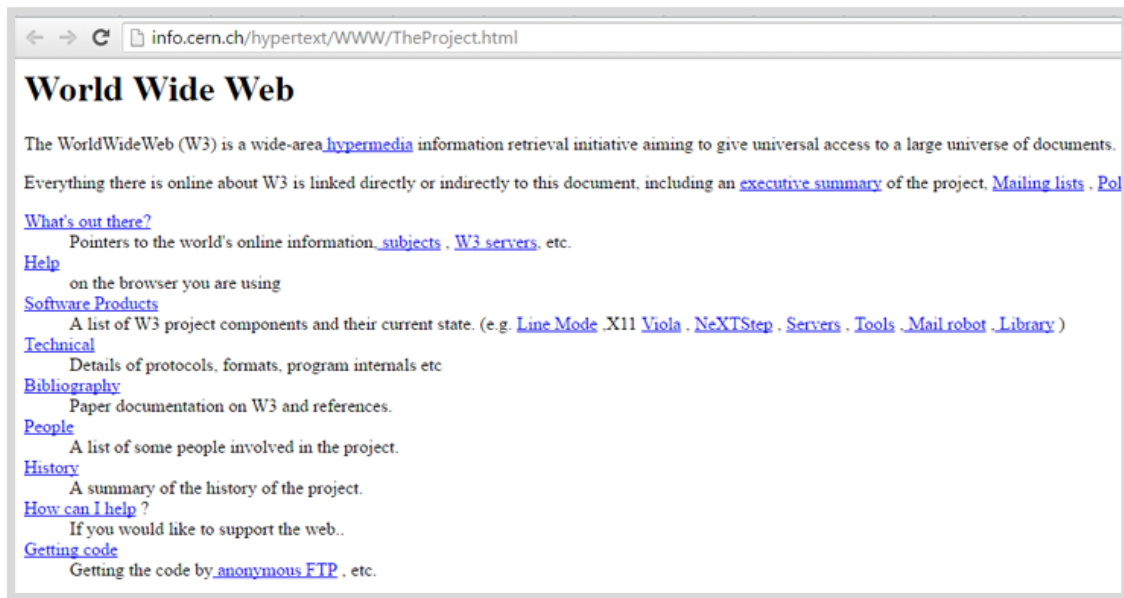
## DATABASE DIAGRAM

Hieronder een overzicht van de verschillende tables in de onderliggende database. De door Entity Framework gegenereerde tables die niet worden gebruikt & bijgevolg geen data bevatten, alsook de table die voor het beheer van de Migration history wordt bijgehouden, zijn niet opgenomen in dit diagram.





Op de dag van dit schrijven, vierde het wereldwijde web zijn 30e verjaardag. Bij zijn ontstaan werden we verwelkomd in een wereld met ogenschijnlijk geen limiet op zijn eigen potentieel. Maar de eerste webpagina's zagen er niet bepaald sexy uit.



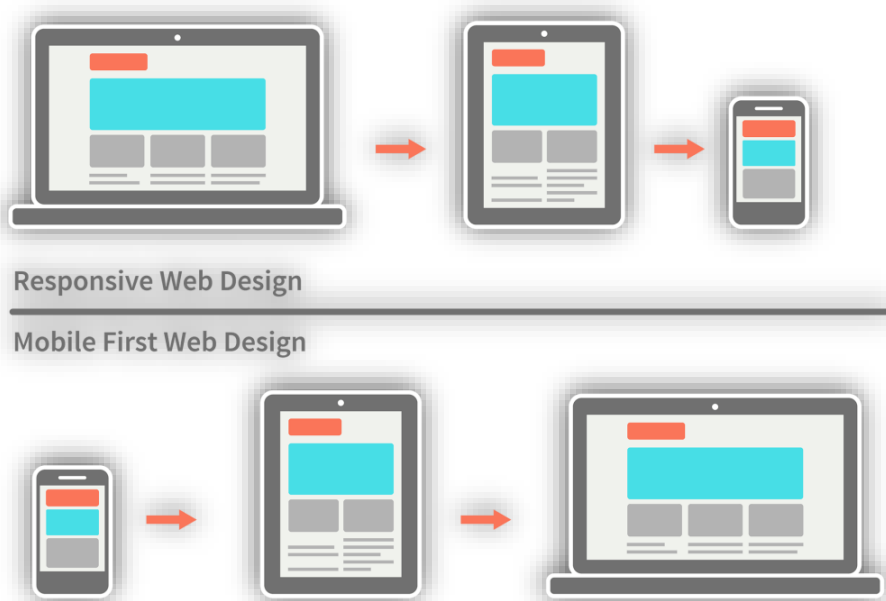
De vraag naar gebruiksvriendelijke en mooie websites kwam er toen steeds meer mensen hun weg vonden naar het wereldwijde web. Dit creëerde nieuwe uitdagingen. Naast de inhoud werd nu ook de lay-out belangrijk. En er veranderde nog meer. De manier waarop we vandaag websites bezoeken, is niet dezelfde als 30 jaar geleden. Waar websites vroeger voornamelijk via desktops of laptops werden bezocht, zien we vandaag hoe smartphones en tablets deze rol overnemen.

Als antwoord op deze trends, is men via heel wat tussenhaltes gekomen tot 'responsive webdesign'. Websites oogden nu fris en modern en waren vlot toegankelijk vanop ieder device. Maar er was ruimte voor verbetering. Men ging er namelijk van uit dat websites vooral via een device met groot scherm werden bezocht. De lay-out van een webpagina werd bijgevolg berekend op de dimensies van een groot scherm. Pas als de website zou detecteren dat ze werd geraadpleegd via een klein scherm, zou de lay-out worden herberekend op basis van de dimensies van dit scherm.

Een jammere zaak omwille van twee redenen:

1. Door de opkomst van smartphones, worden steeds meer websites bezocht door middel van devices met klein scherm.
2. Smartphones of tablets hebben doorgaans minder rekenkracht als desktops of laptops.

Een lay-out voor een groot scherm wordt dus vaak onnodig berekend. Als antwoord hierop, ontstond de 'Mobile First' filosofie waarbij devices met kleiner scherm net bevoorreed worden. De lay-out wordt in eerste instantie berekend op kleinere schermen. Pas als de website detecteert dat een device met groot scherm de website raadpleegt, wordt de lay-out herberekend naar een groter formaat. Hierbij kan gebruik gemaakt worden van de doorgaans grotere rekenkracht van een device met groot scherm. Men draait de boel dus om.



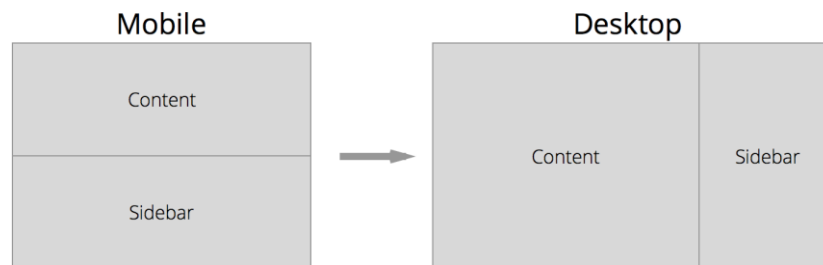
Als we vervolgens in de technische details duiken, moeten we het eerst hebben over media queries. Aan de hand van een media query wordt bepaald welke CSS gebruikt wordt afhankelijk van de dimensies van het scherm. Zo zal de achtergrond van een website door onderstaande code rood kleuren indien deze bezocht wordt door een scherm dat smaller is dan 600px.

```
body {  
  background: green;  
}  
  
// This applies from 0px to 600px  
@media (max-width: 600px) {  
  body {  
    background: red;  
  }  
}
```

Dit is een voorbeeld van 'Desktop First'. Bij 'Mobile First' keert men dit proces om. De CSS zal eerst de lay-out voor een kleiner scherm berekenen. Pas bij een breedte van 600px zal een andere lay-out berekend worden.

```
body {  
  background: red;  
}  
  
// This applies from 600px onwards  
@media (min-width: 600px) {  
  body {  
    background: green;  
  }  
}
```

'Mobile First' biedt ook een voordeel voor de ontwikkelaar. Het is namelijk zo dat men bij het ontwikkelen van een mobiele user interface sneller gebruik kan maken van de default waardes van bepaalde elementen. Stel dat we een website maken waar er naast de hoofdinhoud ook een sidebar is. Het is op een kleiner scherm interessanter om deze sidebar onder de hoofdcontent weer te geven:



In het geval van een kleiner scherm gebruikt de sidebar 100% van de schermbreedte. In het geval van een groter scherm, komt de sidebar naast de hoofdinhoud te staan en zal deze pakweg 40% van het scherm innemen. Bij een 'Mobile First' aanpak kunnen we profiteren van de default waarde van de width property van een <div>:

```
.content {  
  // Properties for smaller screens.  
  // Nothing is required here because we can use the default  
  styles  
  
  // Properties for larger screens  
  @media (min-width: 800px) {  
    float: left;  
    width: 60%;  
  }  
}
```

Bij 'Desktop First' ontstaat er al sneller een situatie waar we een default waarde moeten 'resetten':

```
.content {  
  // Properties for larger screens.  
  float: left;  
  width: 60%;  
  
  // Properties for smaller screens.  
  // Note that we have to write two default properties to make  
  the layout work  
  @media (max-width: 800px) {  
    float: none;  
    width: 100%;  
  }  
}
```

De 'Mobile First' approach bespaart ons hier 2 lijnen code. Als we deze performantie winst kunnen doortrekken naar een complexe website, kunnen we ook hier concluderen dat 'Mobile First' een mooi voordeel biedt.

## EXTENSIONS

---

### ALERTS

---

Doorheen de applicatie wordt gebruik gemaakt van notifications in de vorm van 'Alerts'. Dit is een door Bootstrap aangeboden component om op een gemakkelijke manier aan de eindgebruiker een beknopt stuk informatie aan te bieden. In de EventTracker applicatie worden deze alerts gebruikt om het succes of falen van een gebruikersactie aan te tonen. Een concreet voorbeeld is het inloggen. Als de gebruiker een correct email adres en paswoord aanlevert, zal een succes alert verschijnen. Indien het paswoord verkeerd zou zijn, wordt de gebruiker hier ook op de hoogte van worden gebracht via een ander type alert.

EventTracker maakt gebruik van extensions methods om alerts makkelijk te integreren binnen ASP.NET Core. Deze techniek wordt dieper in detail uitgeschreven op de website trycatchfail.

(<https://www.trycatchfail.com/2018/01/22/easily-add-bootstrap-alerts-to-your-viewresults-with-asp-net-core/>)

Eerst en vooral voorziet men de nodige HTML markup voor het weergeven van Bootstrap alerts. Gezien we deze alerts doorheen de gehele applicatie willen gebruiken, plaatsen we deze markup in de layout view:

```
<div class="alert-@type alert-dismissible fade show" role="alert">
  <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>
  <strong>@title</strong>
  <p>@body</p>
</div>
```

Door gebruik te maken van onderstaande extension methods kunnen we alerts makkelijk en overzichtelijk binnen de controller integreren. Het enige dat we immers hoeven te doen, is een titel & boodschap meegeven samen met het returnobject van een method.

```
public static class AlertExtensions
{
    public static IActionResult WithSuccess(this IActionResult result, string title, string body)
    {
        return Alert(result, "success", title, body);
    }

    public static IActionResult WithInfo(this IActionResult result, string title, string body)
    {
        return Alert(result, "info", title, body);
    }

    public static IActionResult WithWarning(this IActionResult result, string title, string body)
    {
        return Alert(result, "warning", title, body);
    }

    public static IActionResult WithDanger(this IActionResult result, string title, string body)
    {
        return Alert(result, "danger", title, body);
    }

    private static IActionResult Alert(IActionResult result, string type, string title, string body)
    {
        return new AlertDecoratorResult(result, type, title, body);
    }
}

[HttpGet]
[ActionName("Logout")]
public async Task<IActionResult> LogoutAsync()
{
    await _signInManager.SignOutAsync();
    return RedirectToAction(nameof(Login)).WithSuccess("Success", "Logout succesful");
}
```

Via een helperklasse kunnen we duidelijk te maken welk soort alert we wensen:

```
public class AlertDecoratorResult : IActionResult
{
    public IActionResult Result { get; }
    public string Type { get; }
    public string Title { get; }
    public string Body { get; }

    public AlertDecoratorResult(IActionResult result, string type, string title, string body)
    {
        Result = result;
        Type = type;
        Title = title;
        Body = body;
    }

    public async Task ExecuteResultAsync(ActionContext context)
    {
        var factory = context.HttpContext.RequestServices.GetService<ITempDataDictionaryFactory>();

        var tempData = factory.GetTempData(context.HttpContext);
        tempData["_alert.type"] = Type;
        tempData["_alert.title"] = Title;
        tempData["_alert.body"] = Body;

        await Result.ExecuteResultAsync(context);
    }
}
```

Het resultaat van de 'ExecuteResultAsync' method, vangen we tenslotte op in onze HTML markup via deze kleine toevoeging:

```
@{
    var type = (string)TempData["_alert.type"];
    var title = (string)TempData["_alert.title"];
    var body = (string)TempData["_alert.body"];
}
@if (!string.IsNullOrEmpty(type))
{
    <div class="alert-@type alert-dismissible fade show" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <strong>@title</strong>
        <p>@body</p>
    </div>
}
```

Kort samengevat zijn dit de verschillende stappen die de applicatie doorloopt om alerts te laten verschijnen:

1. Een gebruiker voltooit een actie en activeert daarbij een controller action. Die action eindigt met een returnobject waarop een extension method is toegepast. Deze extension method heeft naast het return object ook een titel en boodschap.
2. Het resultaat van de extension methods is een 'AlertDecoratorResult' object dat verantwoordelijk is om het type, de titel en de boodschap van de alert in een TempData object te steken.
3. Dit TempData object wordt door de View gebruikt om de HTML markup aan te vullen.
4. Het Bootstrap framework doet de rest & de alert verschijnt.

Binnen de EventTracker applicatie worden 2 zelfgemaakte validators gebruikt om een 'Timeframe' object te valideren. Een 'Timeframe' object bestaat uit een 'Id', 'Event Date', 'Start' en 'End' property.

- Het DateNotInPastAttribute zorgt ervoor dat wanneer een superuser/admin een evenement aanmaakt of wijzigt, de datum van dat evenement nooit in het verleden kan liggen.

```
public DateNotInPastAttribute()
    : base(typeof(DateTime),
        DateTime.Today.ToShortDateString(),
        DateTime.MaxValue.ToShortDateString())
{
    ErrorMessage = "Date cannot be situated in the past";
}
```

- Het GreaterThanAttribute zorgt ervoor dat de property waarop het wordt toegepast groter moet zijn dan een meegegeven property. In het geval van de EventTracker applicatie wordt deze Attribute toegepast op de 'End' property om zo te verzekeren dat het einduur op een bepaalde datum niet voor het startuur kan liggen.

```
public class GreaterThanAttribute : ValidationAttribute
{
    public GreaterThanAttribute(string otherProperty)
        : base("{0} must be greater than {1}")
    {
        OtherProperty = otherProperty;
    }

    public string OtherProperty { get; set; }

    public string FormatErrorMessage(string name, string otherName)
    {
        return string.Format(ErrorMessageString, name, otherName);
    }

    protected override ValidationResult
        IsValid(object firstValue, ValidationContext validationContext)
    {
        var firstComparable = firstValue as IComparable;
        var secondComparable = GetSecondComparable(validationContext);

        if (firstComparable != null && secondComparable != null)
        {
            if (firstComparable.CompareTo(secondComparable) < 1)
            {
                object obj = validationContext.ObjectInstance;
                var thing = obj.GetType().GetProperty(OtherProperty);
                var displayName = (DisplayAttribute)Attribute.GetCustomAttribute(thing, typeof(DisplayAttribute));

                return new ValidationResult(
                    FormatErrorMessage(validationContext.DisplayName, displayName.GetName()));
            }
        }

        return ValidationResult.Success;
    }

    protected IComparable GetSecondComparable(
        ValidationContext validationContext)
    {
        var propertyInfo = validationContext
            .ObjectType
            .GetProperty(OtherProperty);
        if (propertyInfo != null)
        {
            var secondValue = propertyInfo.GetValue(
                validationContext.ObjectInstance, null);
            return secondValue as IComparable;
        }
        return null;
    }
}
```

## TIMEFRAMES TOEVOEGEN

Een bijzonder complexe functionaliteit van de EventTracker applicatie is het toevoegen van nieuwe evenementen omwille van de 'Timeframe' property. Een evenement heeft enkele simpele properties zoals een naam of een omschrijving. Bij het implementeren van een 'Add Event' functionaliteit, zorgen de MVC TagHelpers, met dank aan .Net Core, voor een doodeenvoudige implementering in een View:

```
<div class="mb-3">
  <label asp-for="Name"></label>
  <input class="form-control" asp-for="Name" />
  <span asp-validation-for="Name"></span>
</div>
<div class="mb-3">
  <label asp-for="Description"></label>
  <textarea class="form-control" asp-for="Description"></textarea>
  <span asp-validation-for="Description"></span>
</div>
```

Deze TagHelpers genereren tijdens het compileren de nodige HTML voor een succesvolle modelbinding:

```
<div>
  <label for="Name">Event Name</label>
  <input id="Name" class="form-control" type="text" data-val="true" data-val-maxlength="The name you've entered is too long" data-val-maxlength-max="50" data-val-required="The Event Name field is required." name="Name" value="" style="background-image: url('data:image/png;base64,iVBORw0KGgoAAA...size: 16px 18px; background-position: 98% 50%; cursor: auto;')>
  <span class="field-validation-valid" data-valmsg-for="Name" data-valmsg-replace="true"></span>
</div>
```

Hier wordt niet alleen het "Name" attribuut gegenereerd, hetgeen nodig is voor model binding, ook de "Data Annotations" waarmee de properties van het model zijn gedecoreerd, zijn omgezet naar de nodige attributen voor validatie.

```
[Required]
[Display(Name = "Event Name")]
[MaxLength(50, ErrorMessage = "The name you've entered is too long")]
public string Name { get; set; }
```

Maar niet alle properties vallen zo makkelijk te implementeren. Zo is er de "Timeframe" collectie van een evenement. Deze property is een "List" van het complexe type "Timeframe". Één enkele timeframe bestaat uit een datum, start- en einduur. Elke property van een "Timeframe" is ook voorzien van enkele "Data Annotations".

```
public class Timeframe
{
    public int Id { get; set; }

    [Required]
    [DataType(DataType.Date, ErrorMessage = "This date is invalid")]
    [DateNotInPast]
    [Display(Name = "Event Date")]
    public DateTime? EventDate { get; set; }

    [Required]
    [Range(0, 24, ErrorMessage = "Invalid hour")]
    [Display(Name = "Start")]
    public int? Starttime { get; set; }

    [Required]
    [Range(0, 24, ErrorMessage = "Invalid hour")]
    [GreaterThan(nameof(Starttime))]
    [Display(Name = "End")]
    public int? Endtime { get; set; }
}
```

Niet alleen de complexere aard van deze property zorgt voor enkele hersenkronkels. De achterliggende domeinproblematiek bemoeilijkt de zaak. Zo moet een gebruiker bij het aanmaken van een nieuw evenement minstens één 'Timeframe' toevoegen. Hoeveel 'Timeframes' een gebruiker zal toevoegen is ongekend.

Hoe zorgen we ervoor dat deze logica gerespecteerd blijft? Hoe kunnen we taghelpers op een dynamische wijze gebruiken zodat elke toegevoegde 'Timeframe' correct zal modelbinden?

De oplossing bestaat uit meerdere stappen. Eerst maken we een Partial View met taghelpers voor de properties van een 'Timeframe':

```
@model Timeframe
<li class="pb-2">
  @using (Html.BeginCollectionItem("Timeframes"))
  {
    <div>
      <div class="form-row align-items-center mb-3">
        <div class="col-sm-4 col-md-4 col-xl-2">
          <label asp-for="EventDate">Date</label>
          <input class="form-control" asp-for="EventDate" />
        </div>
        <div class="col-sm-3 col-md-3 col-xl-2 mt-sm-0 mt-2">
          <label asp-for="Starttime">Start</label>
          <input class="form-control" asp-for="Starttime" min="0" max="24" />
        </div>
        <div class="col-sm-3 col-md-3 col-xl-2 mt-sm-0 mt-2">
          <label asp-for="Endtime">End</label>
          <input class="form-control" asp-for="Endtime" min="0" max="24" />
        </div>
        <div class="col-sm-1 col-md-1 mt-4 mb-md-0 mb-2">
          <button type="button" href="#" class="btn-delete-timeframe btn-delete-md btn-danger form-control"
            data-toggle="tooltip"
            title="Remove timeframe">
            <i class="fa fa-remove"></i>
          </button>
        </div>
      </div>
      <div><span asp-validation-for="EventDate"></span></div>
      <div><span asp-validation-for="Starttime"></span></div>
      <div><span asp-validation-for="Endtime"></span></div>
    </div>
  }
</li>
```

Merk op dat we hier gebruik maken van een 'Html.BeginCollectionItem' helper. Dit is een HtmlHelper die ervoor zorgt dat ook collection properties correct binden. Meer details over deze helper vind je in het hoofdstuk "[Technologieën – Back End](#)".

Deze Partial View integreren we in de 'Add Event' View.

```
<ul id="timeframes" class="list-unstyled pl-0">
  @foreach (Timeframe timeframe in Model.Timeframes)
  {
    <partial name="Partials/_Timeframes" model="timeframe" />
  }
</ul>
<a class="btn btn-secondary mb-3" href="#" id="addTimeframe">Add Timeframe</a>
```

De applicatie loopt hier door de 'Timeframe' collectie van een nieuw evenement. Je zou denken dat daar geen enkel element in zit omdat het om een nieuwe instantie van een evenement gaat. De 'TimeFrame' collectie zou dus leeg moeten zijn. Maar als er geen enkele 'Timeframe' in de collectie zou zitten, zou er ook geen enkel inputveld verschijnen. Pas na het klikken op de knop 'Add Timeframe' zou dergelijke UI verschijnen. Jammerlijk klikwerk omdat we willen dat een gebruiker sowieso 1 'Timeframe' toevoegd. Een perfect moment om dit af te



dwingen en onmiddellijk enkele default waardes mee te geven. We doen dit door in de controller een nieuw 'Event' model aan te maken en onmiddellijk een 'Timeframe' toe te voegen aan de collectie:

```
[HttpGet]
[Authorize(Roles = "Super")]
public IActionResult AddEvent()
{
    var eventToAdd = new Event();
    eventToAdd.Timeframes.Add(new Timeframe { EventDate = DateTime.Today, Starttime = 10, Endtime = 17 });
    return View(eventToAdd);
}
```

Dit alles resulteert in onderstaande user interface:

**Add new event**

**General Info**

Event Name

Description

City Province

Required amount of participants

1

**Timing**

Date Start End

09 / 06 / 2020 10 17

Add Timeframe

Cancel Add

Als we op de 'Add Timeframe' knop duwen, zal er echter niet veel gebeuren. We moeten er immers nog voor zorgen dat de applicatie steeds een Partial View van een 'Timeframe' genereerd telkens we op deze knop klikken. Daarbovenop moeten we verhinderen dat een gebruiker alle 'Timeframes' kan verwijderen en dus een evenement aanmaakt zonder 'Timeframes'.

Om dit probleem om te lossen, voorzien we eerst en vooral een action in de controller die verantwoordelijk is voor het leveren van zo'n Partial View:

```
[HttpGet]
public ActionResult TimeFrameEntry()
{
    return PartialView("Partials/_Timeframes");
}
```

Met behulp van ajax vragen we naar het resultaat van deze action & plakken we hem bij de reeds bestaande HTML:

```
//Add empty timeframe
$("#addTimeframe").click(function () {
    $.get('/Events/TimeFrameEntry', function (template) {
        $("#timeframes").append(template);
        var button = $("#timeframes").children().last().children().find(".btn-delete-timeframe");
        button.click(removeTimeframe);
    });
});
```

De laatste 2 lijnen in bovenstaande JQuery, zorgen ervoor dat bij elke 'Delete' button die gegenereerd wordt, de nodige eventhandler wordt toegevoegd. Deze zorgt ervoor dat er telkens minstens 1 'Timeframe' op de user interface blijft staan.

```
function removeTimeframe() {
    if (document.getElementById("timeframes").getElementsByName("li").length > 1) {
        $(this).parent().parent().parent().parent().remove()
    }
}
```

Uiteindelijk eindigen we met onderstaande user interface:

The screenshot displays the 'Event Tracker 1.0' user interface. It is divided into two main sections: 'General Info' and 'Timing'.

**General Info Section:**

- Event Name:** A text input field with a small icon on the right.
- Description:** A large text area with a small icon on the right.
- City:** A text input field.
- Province:** A text input field.
- Required amount of participants:** A dropdown menu showing the value '1'.

**Timing Section:**

This section contains a table-like structure for adding timeframes. Each row has columns for 'Date', 'Start', and 'End', followed by a red 'X' button to delete the entry.

Date	Start	End	Action
09 / 06 / 2019	10	17	X
10 / 06 / 2019	10	17	X
11 / 06 / 2019	10	17	X

At the bottom of the 'Timing' section is a button labeled 'Add Timeframe'.

In het hoofdstuk '[De Models Layer](#)' werd reeds vermeld dat de entiteit 'Event' niet over een ViewModel beschikt terwijl de entiteit 'UserProfile' over meerdere ViewModels beschikt. Nochtans wordt er doorgaans aangeraden om met ViewModels te werken. De reden hiervoor: overposting! Om te kunnen verduidelijken waarom er bij de entiteit 'Event' niet voor is gekozen om met ViewModels te werken, moet eerst worden uitgelegd wat overposten is en waarom het belangrijk is om een applicatie hiertegen te beschermen.

Overposting is een kwetsbaarheid binnen een webapplicatie waarbij er een onvoldoende afgeschermd datamodel bij een POST operatie teruggestuurd wordt naar de server waardoor een gebruiker in staat is om meer data dan gewenst/toegestaan terug te sturen. Een concreet voorbeeld brengt meer duidelijkheid:

Stel, we hebben onderstaand UserModel:

```
public class UserModel
{
    public string Name { get; set; }
    public bool IsAdmin { get; set; }
}
```

We wensen dat een gebruiker enkel de 'Name' property kan wijzigen. Het zou immers niet wenselijk zijn dat de gebruiker zomaar kan beslissen wie er een admin is en wie niet. We implementeren onderstaande controller met bijbehorend model.

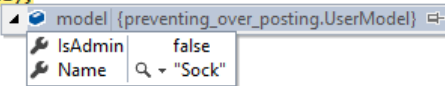
```
[HttpPost]
public IActionResult Vulnerable(UserModel model)
{
    return View("Index", model);
}
```

```
@model UserModel

<form asp-action="Vulnerable" asp-Controller="Home">
    <div class="form-group">
        <label asp-for="Name"></label>
        <input class="form-control" type="TextBox" asp-for="Name" />
    </div>
    <div class="form-group">
        @if (Model.IsAdmin)
        {
            <i>You are an admin</i>
        }
        else
        {
            <i>You are a standard user</i>
        }
    </div>
    <button class="btn btn-sm" type="submit">Submit</button>
</form>
```

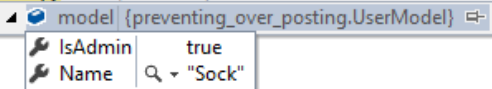
In deze View gebruiken we het UserModel en voorzien we de nodige html waardoor een gebruiker op een eenvoudige en gebruiksvriendelijke manier de 'Name' property kan wijzigen. Als we de applicatie zouden debuggen, krijgen we volgende waardes te zien bij het wijzigen van de 'Name' property via de aangeboden user interface:

```
[HttpPost]
public IActionResult Vulnerable(UserModel model)
{
    return View("Index", model);
}
```



Een kwaadaardige gebruiker kan echter met behulp van tools zoals Postman of Fiddler de 'IsAdmin' property toch manipuleren:

```
[HttpPost]
public IActionResult Vulnerable(UserModel model)
{
    return View("Index", model);
}
```



Dit komt omdat de 'IsAdmin' property nu eenmaal deel uitmaakt van het UserModel. Tevens is de property 'public' dus vrij toegankelijk voor iedereen. Op geen enkele manier wordt er in de code voor gezorgd dat de property niet gewijzigd kan worden door een gebruiker.

Er zijn meerdere manieren om een applicatie te wapenen tegen overposting. Hieronder enkele mogelijkheden:

- Gebruik maken van het [BindAttribute] zorgt ervoor dat men in de controller kan specificeren welke properties er gewijzigd kunnen worden:

```
public IActionResult Safe1([Bind(nameof(UserModel.Name))] UserModel model)
{
    return View("Index", model);
}
```

- Een soortgelijk resultaat kan bereikt worden met de properties van het model te bekleden met het [Editable] of [BindNever] attribuut.

```
public class UserModel
{
    [MaxLength(200)]
    [Display(Name = "Full name")]
    [Required]
    public string Name { get; set; }

    [Editable(false)]
    public bool IsAdmin { get; set; }
}
```

- Veruit de meest gebruikte methode is het gebruik van ViewModels. Dit kan beschouwd worden als gesimplifieerde versies van een entiteit. Deze ViewModels worden gebruikt om data van de client naar de server te sturen en bevatten enkel de properties die mogen worden gemanipuleerd. Zelfs als een kwaadaardige gebruiker de data in het post request zou manipuleren, wordt er met deze data geen rekening gehouden op de server.

```

public class BindingModel
{
    [MaxLength(200)]
    [Display(Name = "Full name")]
    [Required]
    public string Name { get; set; }
}

public class UserModel
{
    [MaxLength(200)]
    [Display(Name = "Full name")]
    [Required]
    public string Name { get; set; }

    [Editable(false)]
    public bool IsAdmin { get; set; }
}

```

```

public IActionResult Safe3(BindingModel bindingModel)
{
    var model = new UserModel();

    // can be simplified using AutoMapper
    model.Name = bindingModel.Name;

    return View("Index", model);
}

```

Er zijn nog mogelijkheden om overposting te voorkomen. Deze liggen echter buiten de scope van dit project. Indien gewenst kan men hier meer over te weten komen op deze [blogpost](https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/) (<https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/>). Het is tevens deze post waaruit bovenstaande voorbeelden zijn onttrokken.

Binnen de EventTracker applicatie worden geen van bovenstaande technieken noch een andere techniek gehanteerd om overposting op het 'Event' model te voorkomen. Dit voor de simpele reden dat er geen gevaar tot overposten is. Laat ons het 'Event' model even in detail bekijken:

```

public class Event
{
    public Event()
    {
        Timeframes = new List<Timeframe>();
    }

    public int Id { get; set; }

    [Required]
    [Display(Name = "Event Name")]
    [MaxLength(50, ErrorMessage = "The name you've entered is too long")]
    public string Name { get; set; }

    [Required]
    [MaxLength(500, ErrorMessage = "The description you've entered is too long")]
    public string Description { get; set; }

    [Required]
    [Range(1, int.MaxValue, ErrorMessage = "Enter a valid number larger than 0")]
    public int WantedAmountOfParticipants { get; set; }

    public List<Timeframe> Timeframes { get; set; }
    public Location Location { get; set; }
    public ICollection<UserEvents> UserEvents { get; set; }
    public bool IsCancelled { get; set; }
}

```

Iedere property van dit model mag gemanipuleerd worden. Dit is zelfs in grote mate de boedeling van de applicatie voor zowel de superusers als de admins. Niet alleen moeten zij nieuwe evenementen aanmaken, zij zijn ook verantwoordelijk voor deze up to date te houden. Stel dat de datum van een evenement verandert, dan is het de bedoeling dat een superuser/admin dit aanpast via de applicatie. In dat geval zou hij de timeframes collectie benaderen en aanpassen. Ook de 'Name', 'Description', 'WantedAmountOfParticipants', 'Location', 'UserEvents' en 'IsCancelled' property moeten aangepast kunnen worden. De enige property die niet gewijzigd mag worden, is de 'Id' property en gezien deze property door Entity Framework gebruikt wordt als primary key in de onderliggende databank, kan deze niet gewijzigd worden.

Dit alles gaat er wel van uit dat evenementen enkel en alleen aangemaakt en gewijzigd kunnen worden door gebruikers met de daarvoor voorziene autorisatie. Zoals hierboven reeds vermeld, zijn dit dus de superusers en de admins van de applicatie. Deze autorisatie wordt toegewezen aan een 'UserProfile' model via de 'UserRole' property:

```
public class UserProfile : IdentityUser
{
    [Required]
    [Display(Name = "First Name")]
    public string FirstName { get; set; }

    [Required]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }

    public UserRole UserRole { get; set; }
    public ICollection<UserEvents> UserEvents { get; set; }
}
```

Gezien een 'UserProfile' overerft van 'IdentityUser', wordt dit model gebruikt bij alle functionaliteiten omtrent authenticatie, zoals bijvoorbeeld bij het inloggen:

```
@model UserProfileLoginViewModel
@{
    ViewBag.Title = "Login";
}

<form method="post">
    <div class="form-group">
        <label asp-for="Email">Email</label>
        <input asp-for="Email" class="form-control" />
        <span asp-validation-for="Email"></span>
    </div>
    <div class="form-group">
        <label asp-for="Password"></label>
        <div class="input-group">
            <input asp-for="Password" type="password" class="form-control password" />
            <div class="input-group-append">
                <span class="btn btn-outline-secondary toggle-password">
                    <i class="fa fa-eye"></i>
                </span>
            </div>
        </div>
        <span asp-validation-for="Password"></span>
    </div>
    <div class="form-group">
        <div class="form-check">
            <input asp-for="RememberMe" type="checkbox" class="form-check-input mt-2" />
            <label asp-for="RememberMe" class="form-check-label">Remember me</label>
        </div>
    </div>
    <div asp-validation-summary="ModelOnly"></div>
    <div class="form-group">
        <input type="submit" value="Login" name="Login" class="btn btn-primary float-right" />
    </div>
    <div class="form-group">
        <a asp-action="ForgotPassword" class="text-primary h6">Forgot password?</a>
    </div>
</form>
```

In bovenstaande View is echter gebruik gemaakt van een 'UserProfileLoginViewModel' en niet van het oorspronkelijke 'UserProfile' model. Dit om te vermijden dat een user bij het posten van data tijdens het inloggen ook zijn UserRole zou kunnen aanpassen (bijvoorbeeld naar een hoger niveau). Het 'UserProfileLoginViewModel' beschikt namelijk niet over deze property:

```
public class UserProfileLoginViewModel
{
    [Required]
    [Display(Name = "Email")]
    [DataType(DataType.EmailAddress)]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    public bool RememberMe { get; set; }
}
```

Door gebruik te maken van ViewModels wordt het 'UserProfile' model beschermd tegen overposting. Dit in tegenstelling tot het 'Event' model waar dit dus niet nodig is.