

# HKDSE ICT SBA Report

---

- HKDSE ICT SBA Report
- Objective
- Design and Implementation
  - Program Functions Overview
  - Crucial Features
  - Program Functions Detailed
    - Graphical User Interface (GUI)
      - Home Page
      - Navigation System (App Drawer)
      - Event Details Page
      - Accounts Page, Login Page and Registration Page
        - Accounts Page (Not Logged In)
        - Login Page
        - Register Page
        - Accounts Page (Logged In)
      - AI ChatBot Help Desk
    - Implementation
      - Overview
      - Dataflow
      - Advantages
      - Pascal HTTP Server
        - Intentional Disabled Features
        - Use of classes
        - Use of Custom APIs
  - Testing and Evaluation
    - Data Validation
      - Server
      - Client
    - Custom Pascal Stack Tracer
      - On
      - Off
  - Learning Process Reflection

## Objective

---

**Event Ticketing System (ETS)** aims to

- provide users with ease when booking and reserving tickets for their favourite events
- provide users with a great **user experience** (UX) with the help of a **graphical user interface** (GUI)
- allow **multi-user support** for booking tickets

The Purpose of ETS is to

- allow users to search and discover for events easily
- allow easy ticketing booking and reservation
- allow event organisers to generate analysis report based on their events

## Design and Implementation

---

### Program Functions Overview

- Show all events
  - events are listed out on the home page of the program
- Registration and Login
  - New users can register
  - Old users can login
- Joining events
  - Participants are able to join events
  - If the event is full, they will automatically be queued on the waiting list, and will automatically be moved to joined when space is available
- Creating Events
  - Organisers can create events by the help of a form
- Amendments of events
  - Organisers can edit event details later via the Created Events Page

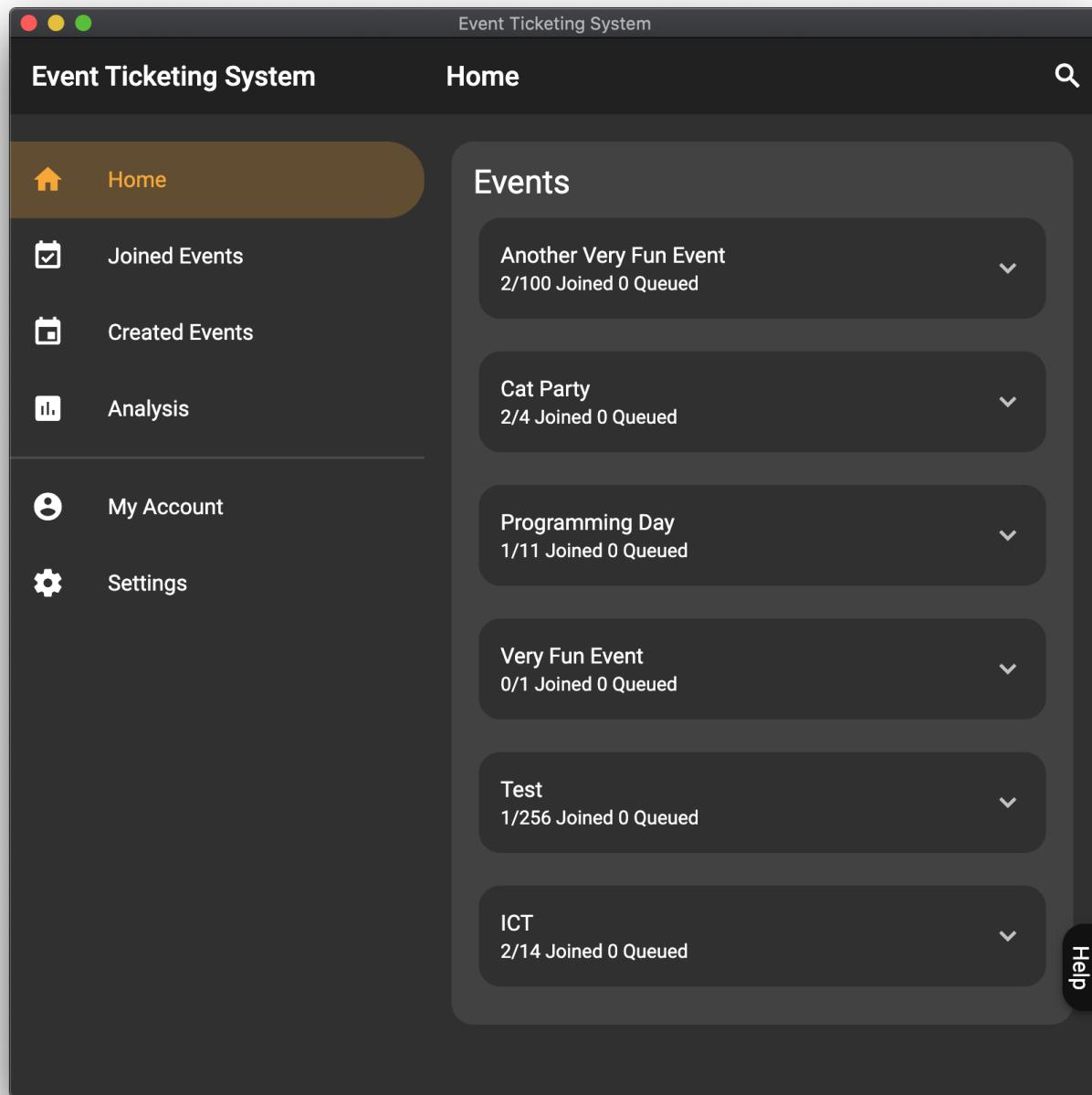
### Crucial Features

- Database
  - Database used in ETS is one custom made from scratch, using a **NoSQL** document-orientated data structure
  - Database files are stored as **json** files
- User Authentication
  - Users are authenticated using a username or email with a password
  - Passwords are not stored directly inside the custom database system, passwords are
    1. first **salted**
    2. and then **hashed**
  - Server authenticate users by salting and hashing input password and compared to the original salted password
  - This uses various concepts in cryptography
- Custom Stack Tracer
  - Since Pascal doesn't really support runtime stacktrace, the custom stack tracer allows developers to view where went wrong easily

### Program Functions Detailed

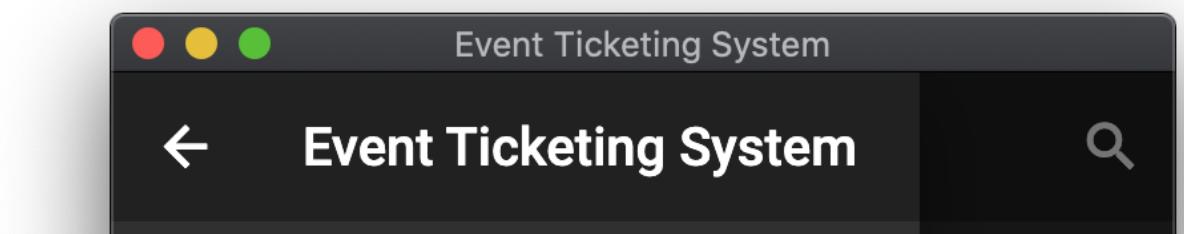
#### Graphical User Interface (GUI)

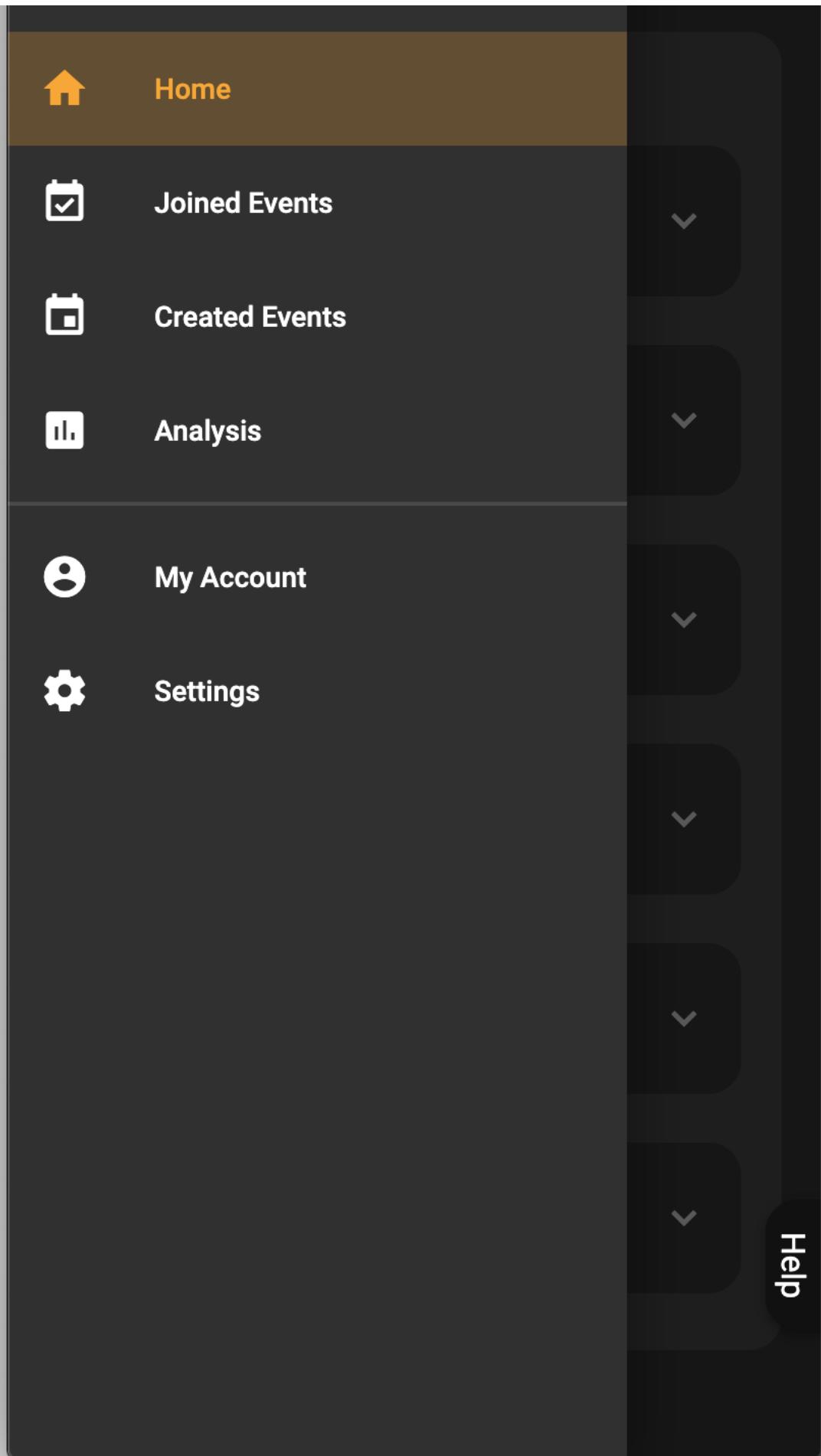
## Home Page

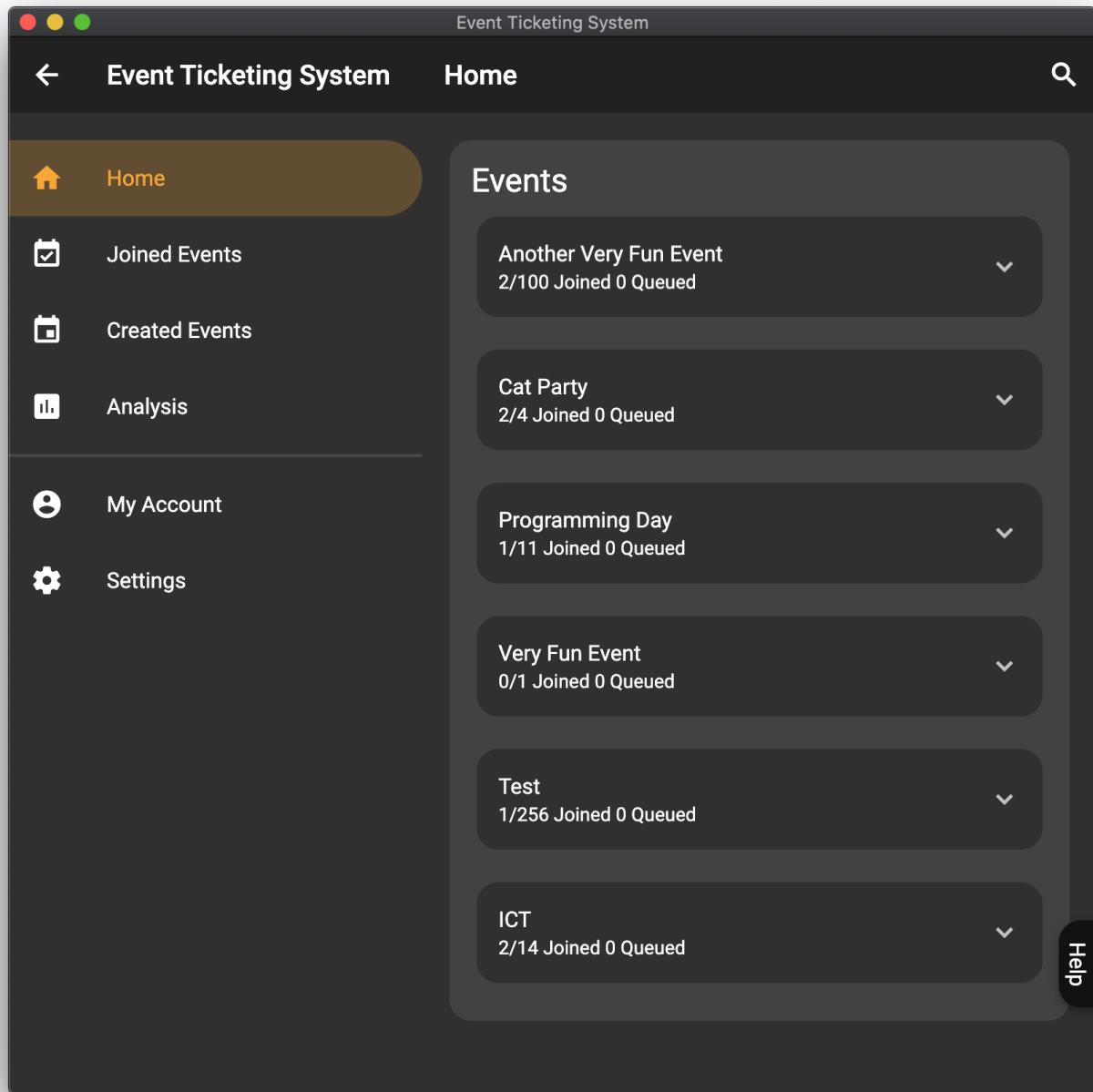


## Navigation System (App Drawer)

- App drawer
  - is present on the left side of the screen for non-mobile users
  - is accessible via the menu icon on the top left of the screen for mobile users







## Event Details Page

This page shows event details and allow participants to select tickets and join



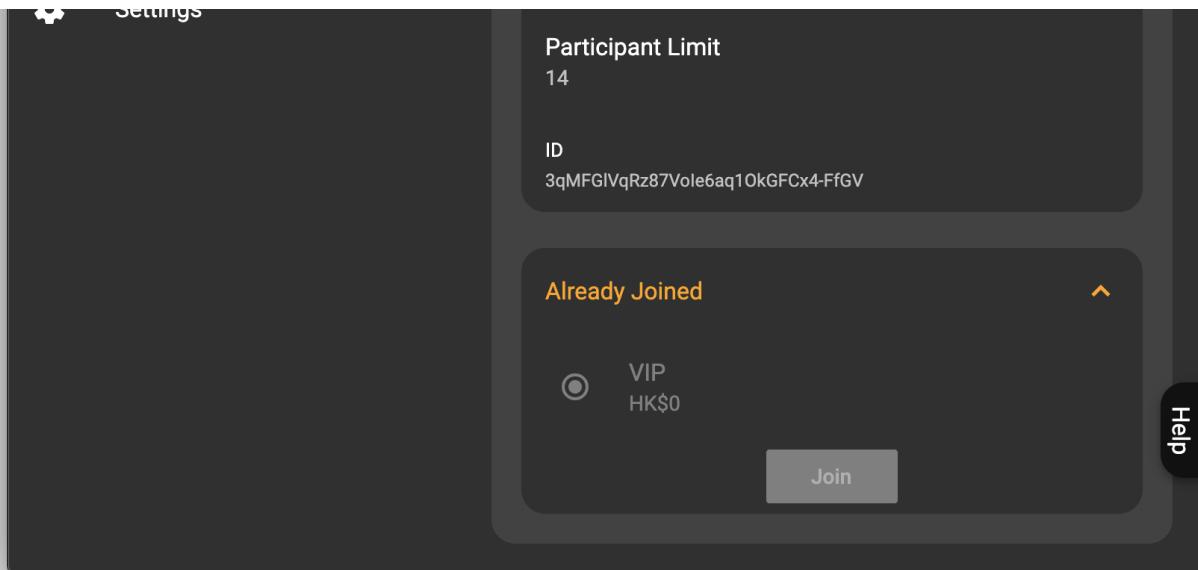
The screenshot shows a dark-themed application window titled "Event Ticketing System". At the top left is a back arrow, and at the top right is a share icon. The main title "Event Details" is centered above a large, rounded rectangular panel. This panel contains the following information:

Event Information	
Name	ICT
Description	Summer Tutorial
Organiser	kmwong
Start	August 26, 2020 9:35:00 AM
End	August 26, 2020 11:55:00 AM
Venue	Zoom
Theme	Core CH.18-19
Participant Limit	14
ID	3qMFGIVqRz87Vole6aq1OkGFCx4-FfGV

To the left of this panel is a sidebar with the following menu items:

- Home
- Joined Events
- Created Events
- Analysis
- My Account
- Settings

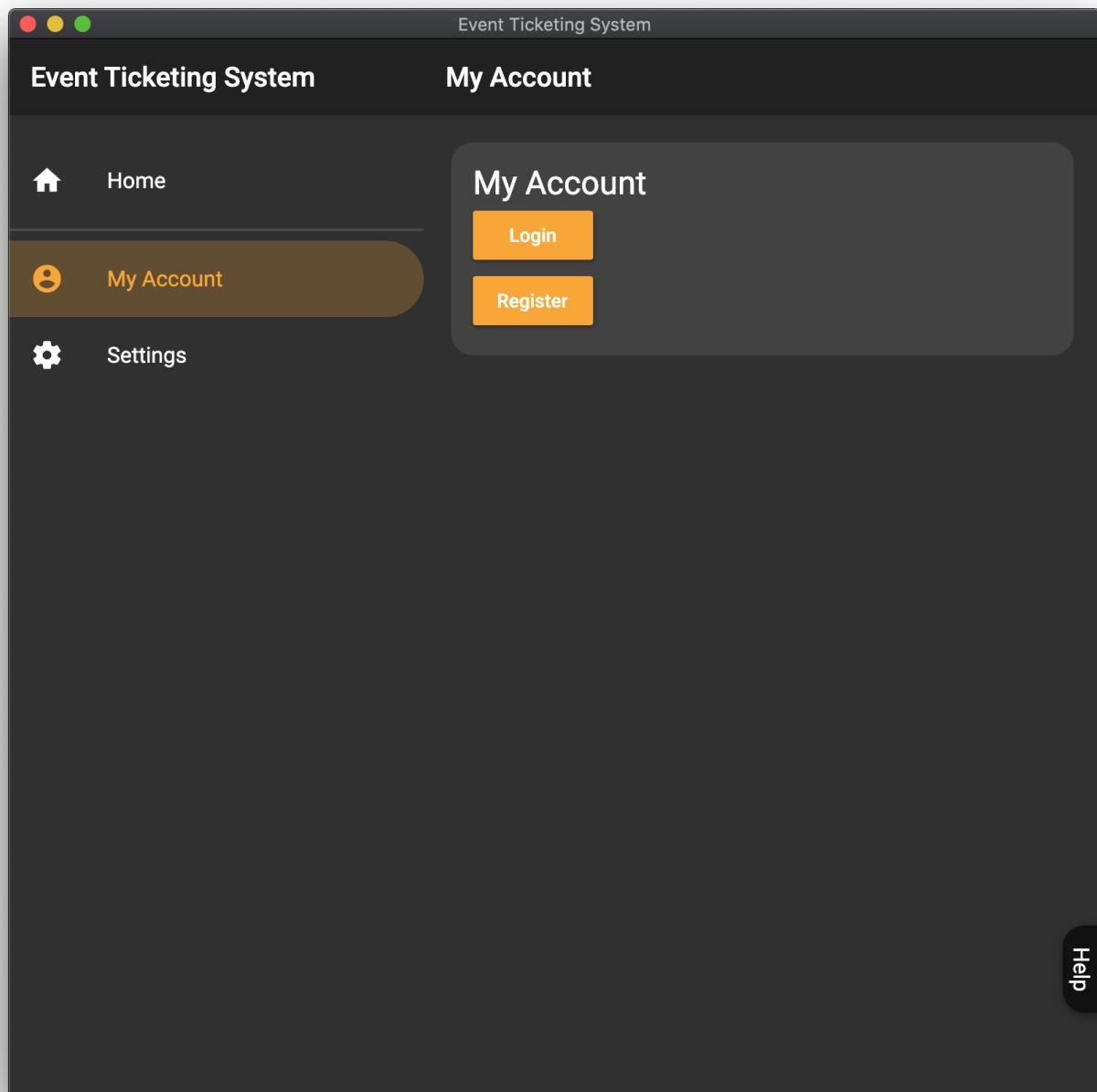
This screenshot is identical to the one above, showing the same "Event Details" panel for the event "ICT". The sidebar on the left also contains the same menu items.



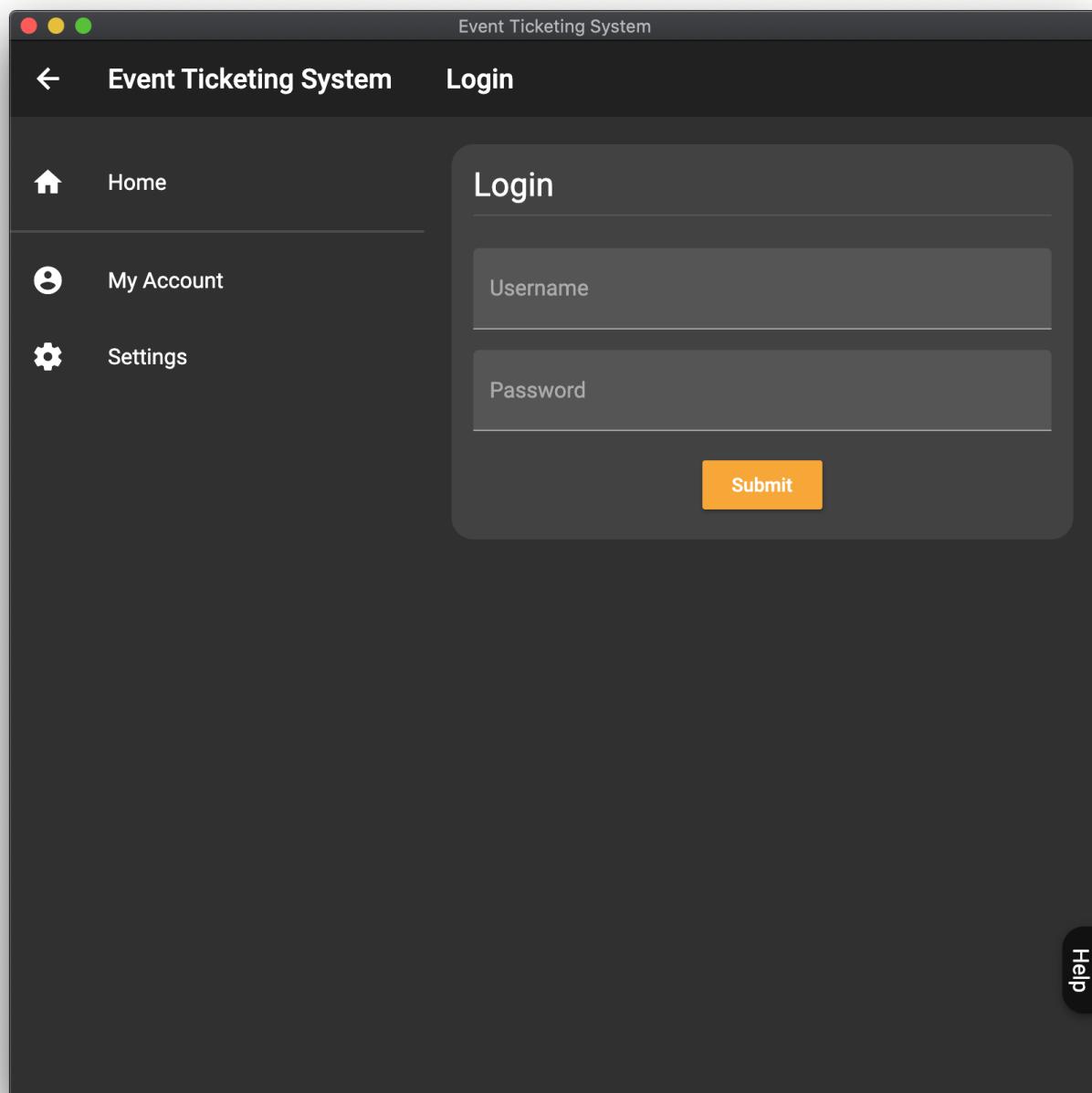
## Accounts Page, Login Page and Registration Page

Accounts page lets users view their account details if logged in, else, login and register button is shown

### Accounts Page (Not Logged In)



**Login Page**



## Register Page

The screenshot shows the 'Register' page of the Event Ticketing System. The page has a dark theme with light-colored input fields. The navigation bar at the top includes a back arrow, the system name 'Event Ticketing System', and the page title 'Register'. On the left, there's a sidebar with links for 'Home', 'My Account', and 'Settings'. The main content area is titled 'Register' and contains the following fields:

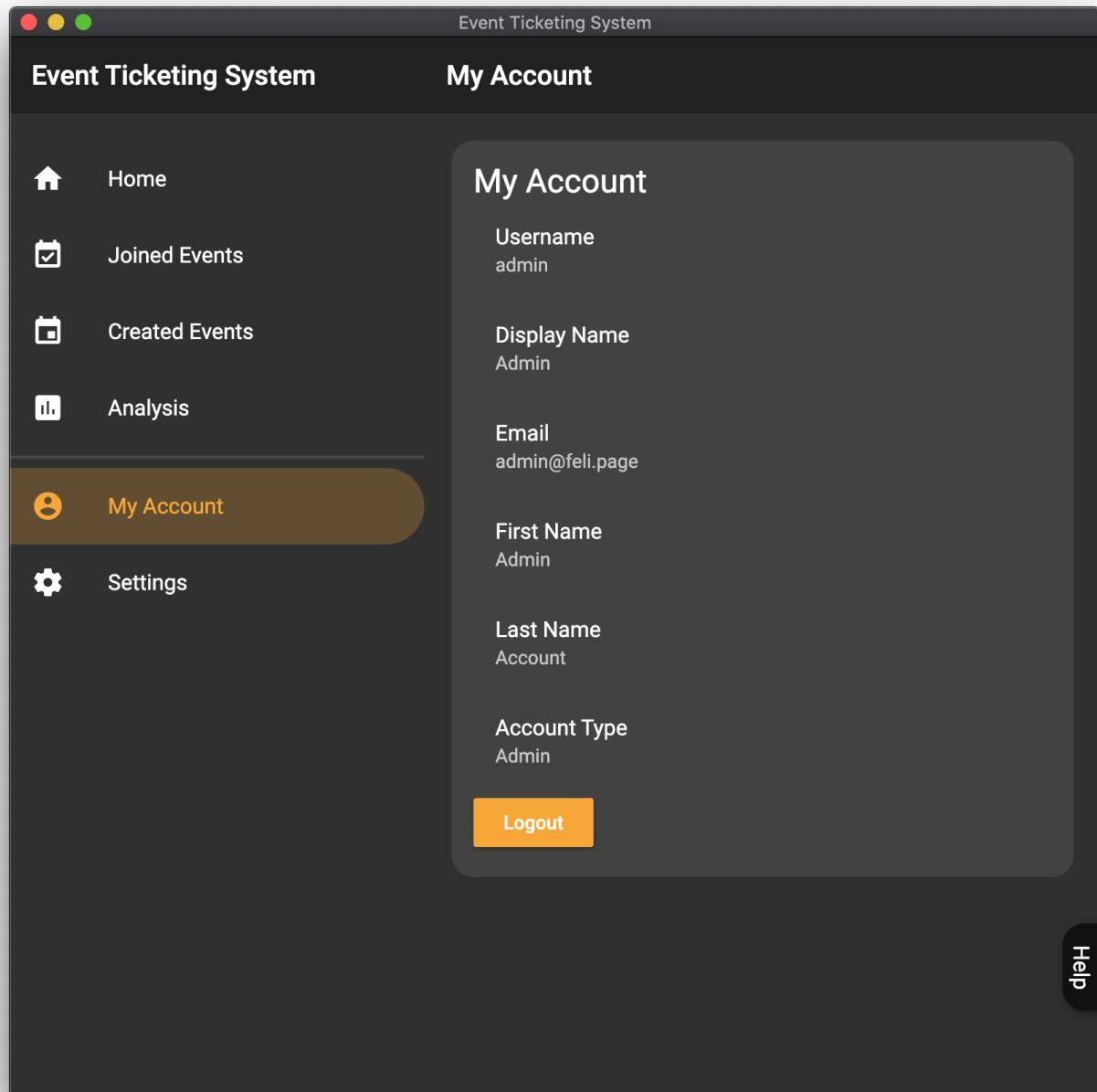
- First Name**: This field is required.
- Last Name**: This field is required.
- Username**: This field is required.
- Email**: `not.an@email`. Error message: Invalid Email.
- Password**: `....`. Error message: Password must have 8 - 32 char...
- Confirm password**: Error message: Those passwords didn't match, ...

Below these fields is a section titled 'Account Type' with two radio button options:

- Participator**
- Organiser**

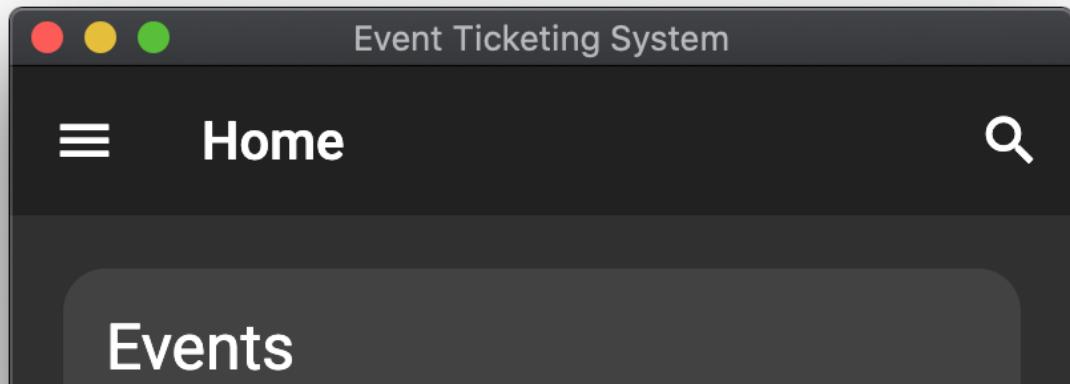
A large orange 'Submit' button is located at the bottom right of the form area. A small 'Help' link is visible on the far right edge of the page.

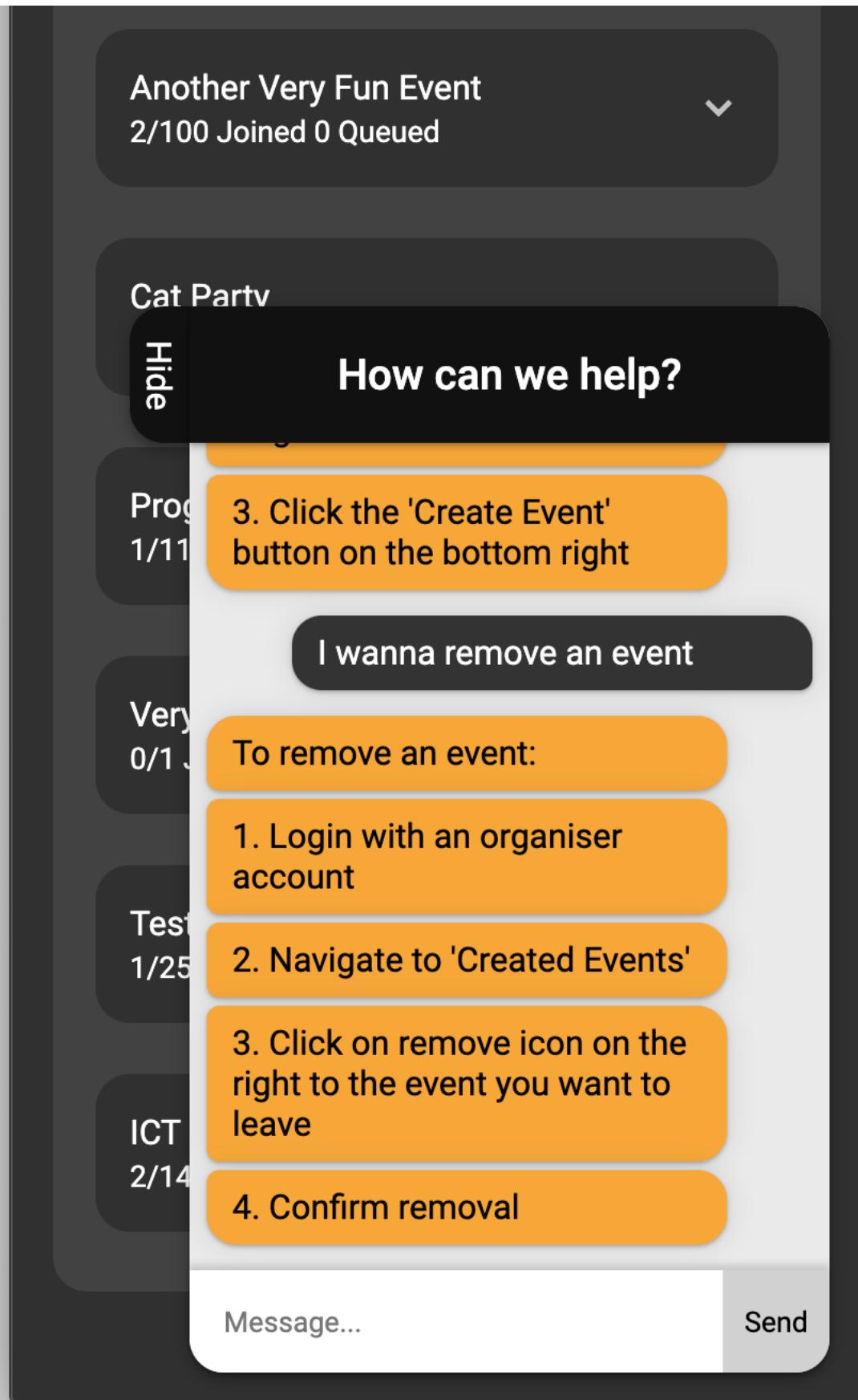
Accounts Page (Logged In)



## AI ChatBot Help Desk

This chat screen allows users to ask help from an AI ChatBot. This ChatBot uses machine learning to map user's dynamic questions to fixed answers.





## Overview

Event Ticketing System consist of two parts

1. Server (Backend)
  - ETS Rest API (Written in `objectpascal`)
  - ChatBot Rest API (Written in `javascript`)
  - HTTP & HTTPS Web/Proxy Server (Written in `javascript`)
2. Client (Frontend)
  - Web (Written in `dart`, `html`, `scss`, `javascript`)
  - Android (Written in `dart`)
  - Windows (Written in `javascript`)
  - MacOS (Written in `javascript`)

## Dataflow

1. User request responses from HTTP & HTTPS Web/Proxy Server
2. Server responses with resources

## Advantages

- Most ICT SBA Projects doesn't support multi-user since the program is only available to one user, if copied to another machine, they won't share the same resources
- Our Event Ticketing System uses client and server concepts so multiple clients can connect to the same server, allowing multi-user support
- Since our server is based on Rest APIs with `cross origin` enabled, any other developers can use the API to create their own frontend with our backend server

## Pascal HTTP Server

### **Intentional Disabled Features**

To prevent double booking, the ETS Rest API HTTP Server has multi-thread disabled, i.e. one request at a time.

### **Use of classes**

On the server, all everything queried from the custom database are collections of documents or just documents. Users, Events, Tickets, etc are classes extending from collections, while User, Event, Ticket, etc are classes extending from documents.

This prevents double written codes, all Users, Events, Tickets, etc inherits functions from the collection bases class, same applies for User, Event, Ticket, etc

### **Use of Custom APIs**

1. FeliStorageAPI
  - All queries from database, the code looks nicer and easier to maintain
2. FeliFileAPI

- All reading and writing of files is simplified to just using this API

# Testing and Evaluation

---

## Data Validation

### Server

All new users and events are validated before inserting into database to prevent invalid requests

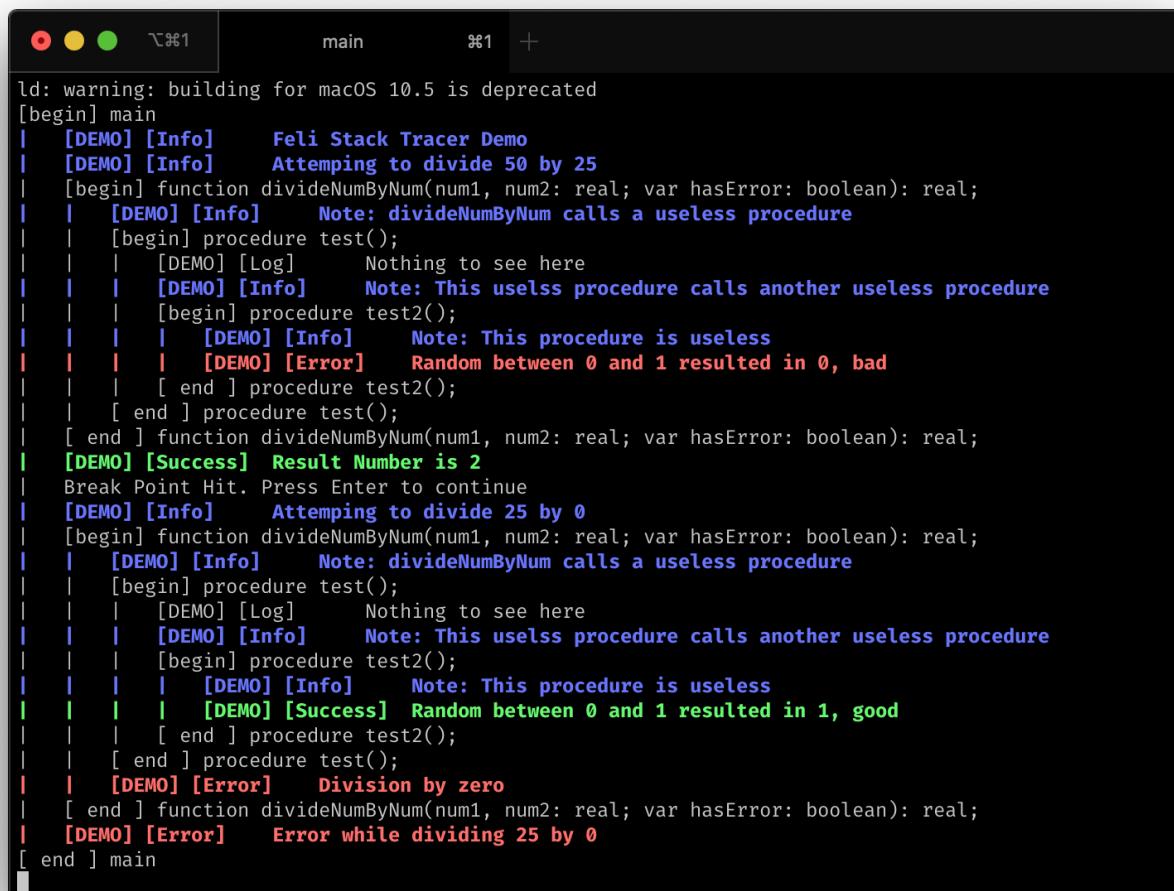
### Client

TextFields will automatically validate when submit buttons are pressed. Respective errors are displayed directly below the TextField for easy reading

## Custom Pascal Stack Tracer

This custom build stack tracer allows developers to trace procedures and functions easily during development

### On

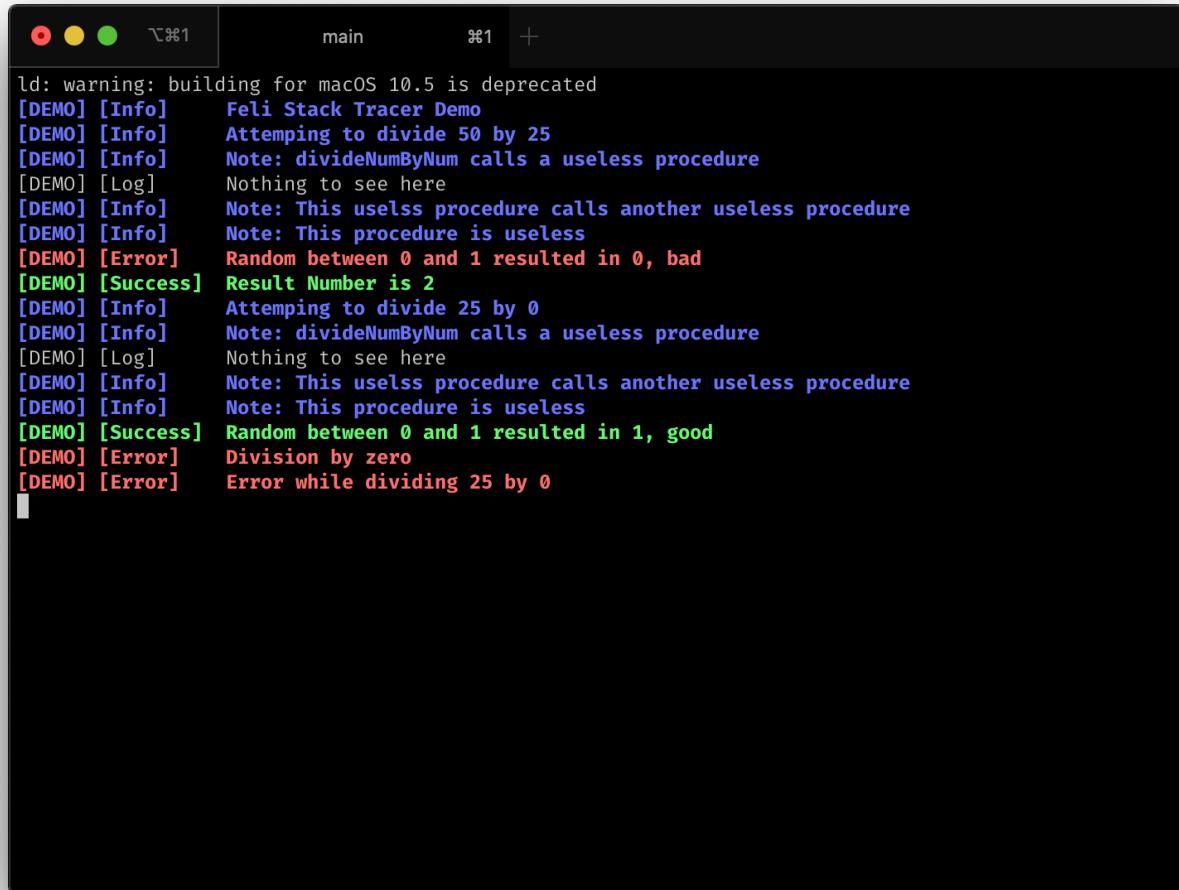


```

ld: warning: building for macOS 10.5 is deprecated
[begin] main
| [DEMO] [Info]      Feli Stack Tracer Demo
| [DEMO] [Info]      Attempting to divide 50 by 25
| [begin] function divideNumByNum(num1, num2: real; var hasError: boolean): real;
| | [DEMO] [Info]      Note: divideNumByNum calls a useless procedure
| | [begin] procedure test();
| | | [DEMO] [Log]      Nothing to see here
| | | [DEMO] [Info]      Note: This uselss procedure calls another useless procedure
| | | [begin] procedure test2();
| | | | [DEMO] [Info]      Note: This procedure is useless
| | | | [DEMO] [Error]    Random between 0 and 1 resulted in 0, bad
| | | [end] procedure test2();
| | [end] procedure test();
| [end] function divideNumByNum(num1, num2: real; var hasError: boolean): real;
| [DEMO] [Success]  Result Number is 2
Break Point Hit. Press Enter to continue
| [DEMO] [Info]      Attempting to divide 25 by 0
[begin] function divideNumByNum(num1, num2: real; var hasError: boolean): real;
| | [DEMO] [Info]      Note: divideNumByNum calls a useless procedure
| | [begin] procedure test();
| | | [DEMO] [Log]      Nothing to see here
| | | [DEMO] [Info]      Note: This uselss procedure calls another useless procedure
| | | [begin] procedure test2();
| | | | [DEMO] [Info]      Note: This procedure is useless
| | | | [DEMO] [Success] Random between 0 and 1 resulted in 1, good
| | | [end] procedure test2();
| | [end] procedure test();
| | [DEMO] [Error]    Division by zero
| [end] function divideNumByNum(num1, num2: real; var hasError: boolean): real;
| [DEMO] [Error]    Error while dividing 25 by 0
[ end ] main

```

### Off



The screenshot shows a terminal window titled "main" with the command "⌘1". The window contains the following text:

```
ld: warning: building for macOS 10.5 is deprecated
[DEMO] [Info]    Feli Stack Tracer Demo
[DEMO] [Info]    Attempting to divide 50 by 25
[DEMO] [Info]    Note: divideNumByNum calls a useless procedure
[DEMO] [Log]     Nothing to see here
[DEMO] [Info]    Note: This useless procedure calls another useless procedure
[DEMO] [Info]    Note: This procedure is useless
[DEMO] [Error]   Random between 0 and 1 resulted in 0, bad
[DEMO] [Success] Result Number is 2
[DEMO] [Info]    Attempting to divide 25 by 0
[DEMO] [Info]    Note: divideNumByNum calls a useless procedure
[DEMO] [Log]     Nothing to see here
[DEMO] [Info]    Note: This useless procedure calls another useless procedure
[DEMO] [Info]    Note: This procedure is useless
[DEMO] [Success] Random between 0 and 1 resulted in 1, good
[DEMO] [Error]   Division by zero
[DEMO] [Error]   Error while dividing 25 by 0
```

## Learning Process Reflection

---

Notes to self: Screenshots are captured at 800x800/400x800 pixels with device pixel ratio of 1.0