Design Deliverable

# Team Juan

**Team Website:** https://dhillp.github.io/TeamJuan/

**Members:** Tung Kong, Pamaldeep Singh Dhillon, Kevin T Nguyen, Cynthia Tran, Harmandeep Singh

**Contact us**: official.teamjuan@gmail.com

## Introduction

Our design strives to satisfy the JustBeweave requirements while maintaining a conservative interface. There are different functionalities for the candidate, judge, and admin/contest sponsor to allow for an intuitive, simple, and working program. The GUI creates a smooth transition from the login screen, to the register screen, to the event/schedule list, and so forth. All of the candidate's personal information and the event list will be stored in a text file serving as a database and can be later accessed or modified by the admin or judge.

## Table of Contents

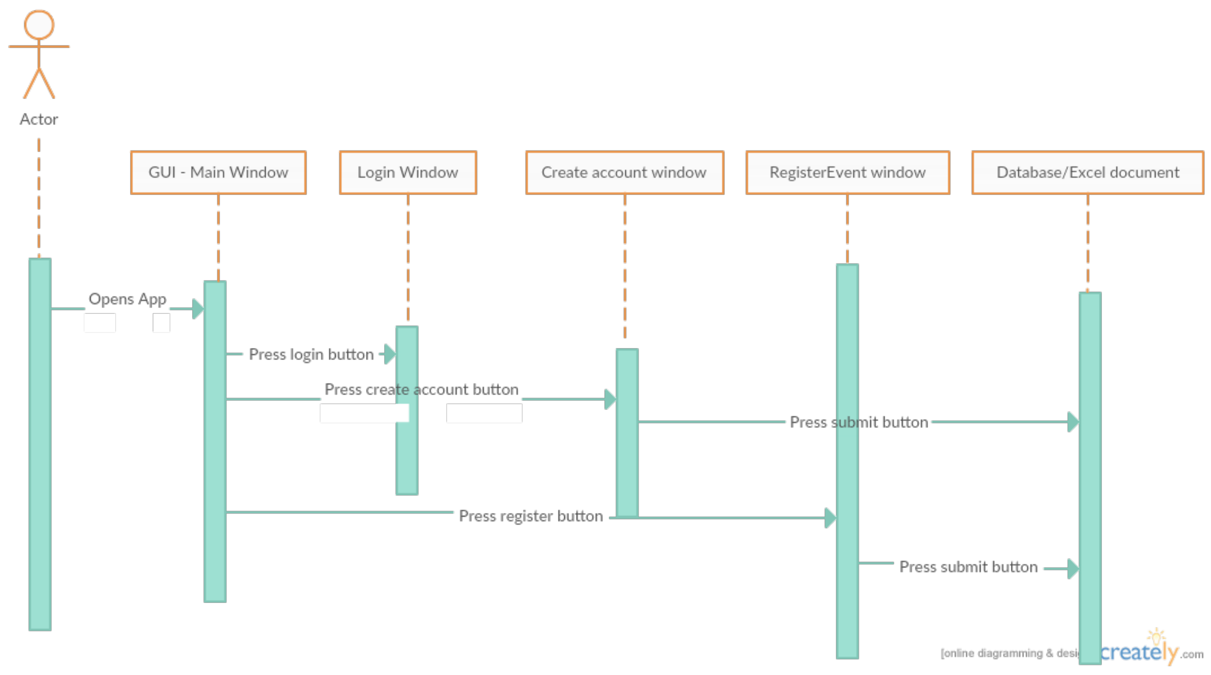
**Rationale Summary:**

For Team Juan's rationale summary we'll talk about the classes that are involved in the backend code before getting the front end GUI code. Our design are based off classes we have decided to implement: **Event Registration, Admin, System Registration,** and **Person** (Judge, Attendee, Contestant). The relationship between the subclasses of Person is that they inherit most of the variables in Person which is why we decided to make the subclasses to be a is-a relationship instead of having the classes to have unique data variables that are essentially that same data being duplicated. The Event Registration class serves the event registration process by holding a person's ID, which is entered and the event that the user has wished to enter by selecting the event and category functions. The Admin class is reserved for one user that could manage the events that are being stored in a database structure. The data is updated if any changes to any events are made. We are still deciding whether the admin class should be pre implemented or chosen when the application is ran. The System Registration has the ability to contact the Person class by which it creates a Person. The Person class hold data consisting of the information that are given from a user, firstname, lastname, email, and category. The ID is generated from a random generator that gives the user it's unique code which could applied to the event registration class. The classes that are under the Person class serves as the distinction between certain applicants, since we wanted to make the getlist function inside each of the classes refers to it's respectable group.

For the GUI end, we decided to implement the GUI classes: **Create Account Screen, Login screen, Register Screen, Events List Screen, and Judge Schedule List Screen.** The title of each of the classes are straightforward. The login screen present a login screen that has the ability to let a user login or create an account if they don't. If they don't have an account, the create account screen displays after the register for account button is chosen. Events list screen and Judge schedule list screen are revealed if their respective buttons are chosen on the login screen. After Login screen, the register screen display which allows the user to register for certain events.

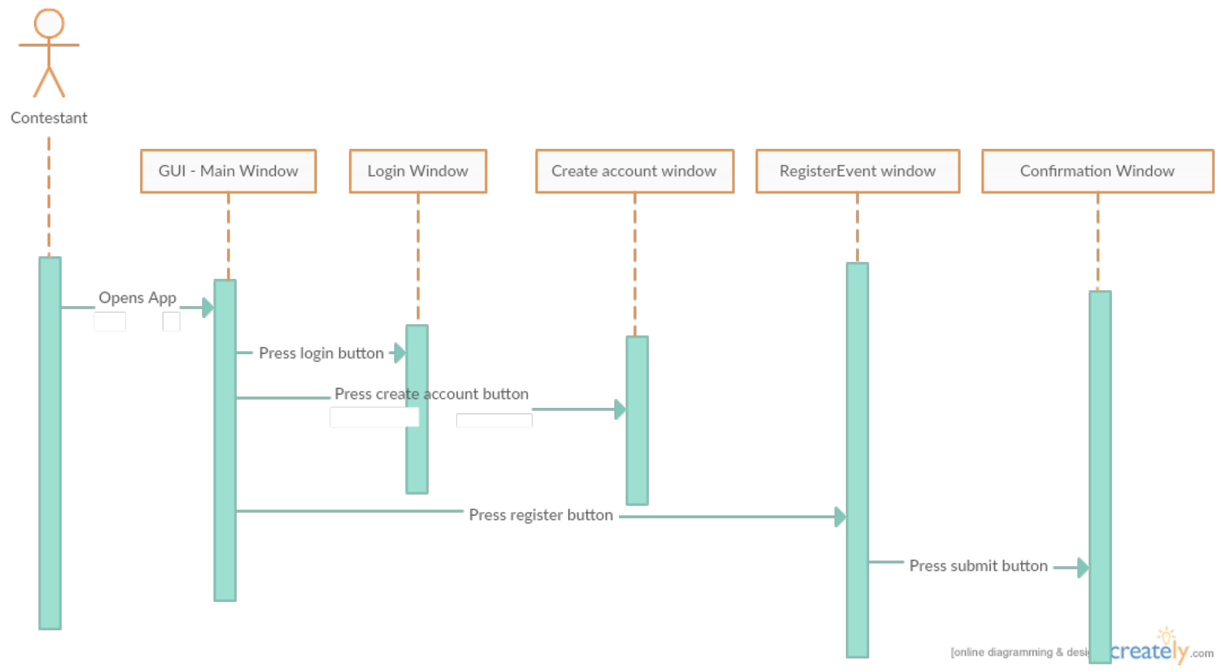
**User Story Sequence Diagrams:**

*As the contest sponsor I want to be able to keep very careful records of contestants who register and the designs they submit, so that I can keep track of it all.*



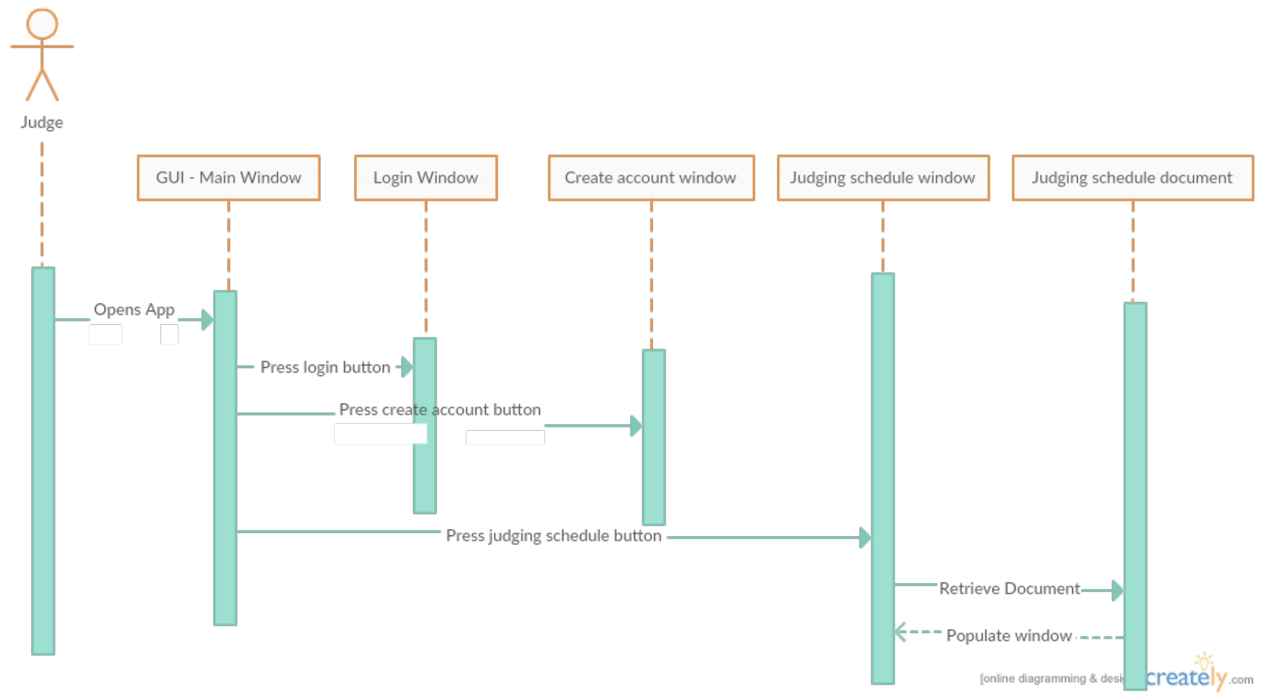
As a sponsor, each time a contestant or judge creates account or registers for an event, their information will be saved within a database/excel sheet.

*As a contestant I want to be able to register so that I can be entered into the contest.*



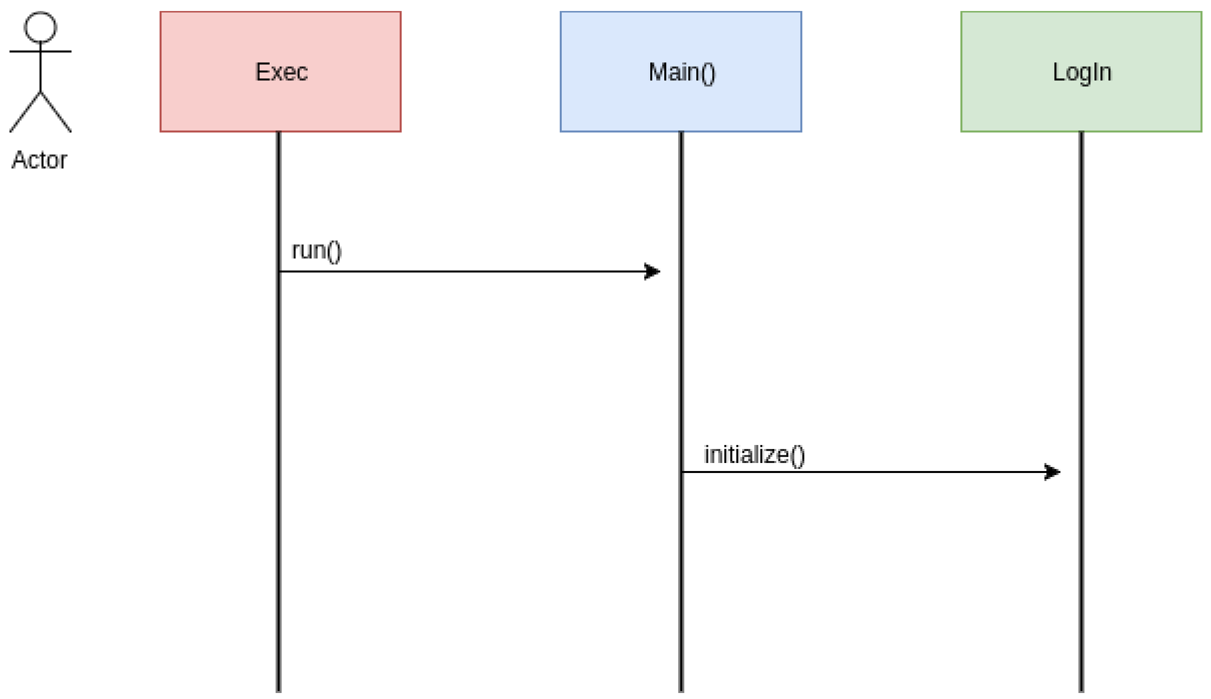
The contestant opens the program which leads to the main GUI frame. If they already have a system account, they can press the login button which opens a separate login window. After that, they can press the register button that opens a separate window so they can register for a certain event. In the register window, they push the submit button and a confirmation window will pop up with details such as entry number. If they don’t have an account, they can press the create account button and this will also open a separate window.

*As a judge I want to be able to see the schedule so I know when I’m judging.*



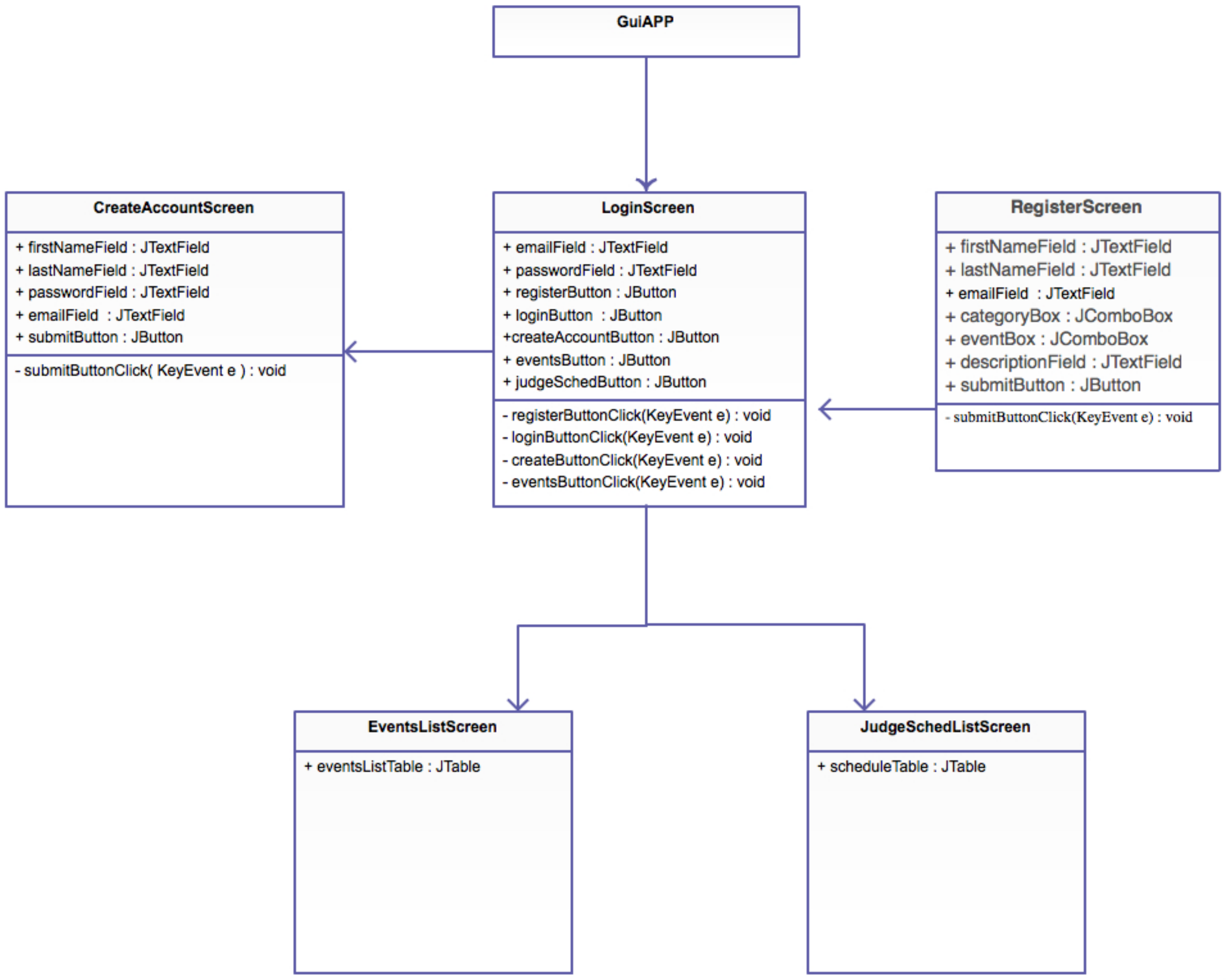
The judge opens the program and if they have an account they press the login button which leads to login screen. Then they push the judging schedule button to open a separate window, which will retrieve the judging schedule document and populate the judging schedule window. If they don’t have an account, they push the create account button to create one.

## System Startup Sequence Diagram:

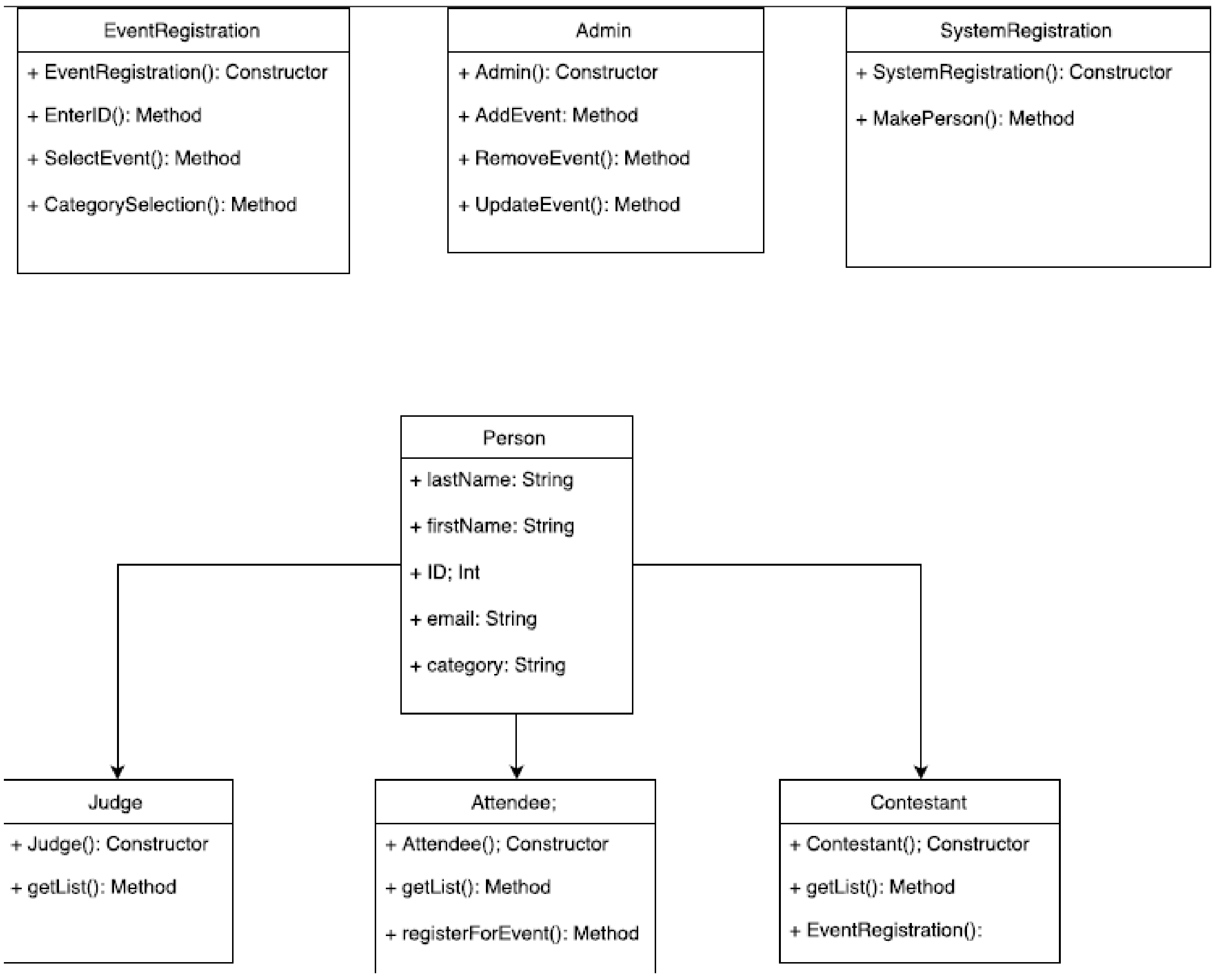


Upon executing the program, the main function initializes the login window and all of the data within it. The login window serves as the catalyst to the other functionalities of the program.

## GUI Class Diagram:


GuiAPP
CreateAccountScreen
+ firstNameField : JTextField
+ lastNameField : JTextField
+ passwordField : JTextField
+ emailField : JTextField
+ submitButton : JButton
- submitButtonClick( KeyEvent e ) : void
LoginScreen
+ emailField : JTextField
+ passwordField : JTextField
+ registerButton : JButton
+ loginButton : JButton
+createAccountButton : JButton
+ eventsButton : JButton
+ judgeSchedButton : JButton
- registerButtonClick(KeyEvent e) : void
- loginButtonClick(KeyEvent e) : void
- createButtonClick(KeyEvent e) : void
- eventsButtonClick(KeyEvent e) : void
RegisterScreen
+ firstNameField : JTextField
+ lastNameField : JTextField
+ emailField : JTextField
+ categoryBox : JComboBox
+ eventBox : JComboBox
+ descriptionField : JTextField
+ submitButton : JButton
- submitButtonClick(KeyEvent e) : void
EventsListScreen
+ eventsListTable : JTable
JudgeSchedListScreen
+ scheduleTable : JTable

## Class Design:


EventRegistration
+ EventRegistration(): Constructor
+ EnterID(): Method
+ SelectEvent(): Method
+ CategorySelection(): Method
Admin
+ Admin(): Constructor
+ AddEvent: Method
+ RemoveEvent(): Method
+ UpdateEvent(): Method
SystemRegistration
+ SystemRegistration(): Constructor
+ MakePerson(): Method
Person
+ lastName: String
+ firstName: String
+ ID; Int
+ email: String
+ category: String
Judge
+ Judge(): Constructor
+ getList(): Method
Attendee;
+ Attendee(); Constructor
+ getList(): Method
+ registerForEvent(): Method
Contestant
+ Contestant(); Constructor
+ getList(): Method
+ EventRegistration():