**ProfileAct**

*Wilfredo Montanez Vazquez<*[*wilfredo.montanez@upr.edu*](mailto:wilfredo.montanez@upr.edu)*>*

*Fernando J Arocho Colom <*[*fernando.arocho@upr.edu*](mailto:fernando.arocho@upr.edu)*>*

*Luis E Millan Martinez<*[*luis.millan3@upr.edu*](mailto:luis.millan3@upr.edu)*>*

**Introduction:**

ProfileAct is a web application software that aims, to benefit students and research groups with their exposure to companies and graduate schools, by providing a simple way of promoting. themselves via an online portfolio. The software will let users upload a profile with their professional information, with the intention of being accessible by recruiters during the job fair week. It will improve the communication between students and potential employers, and facilitate the process of recruitment in campus. It will be open to students from all majors, allowing for a larger group of students to be encouraged to attend the job fair and take advantage of the networking opportunities. Also, it will allow research groups to gain more opportunities for sponsorship from the industry and academia.

This software is going to be developed using Heroku Web services, Sublime 3, Angular.js 1, HTML,CSS, Bootstrap, Node.js and PostgreSQL. The main reason for the software being a web application is so that it is readily available to all users in their browser’s disregarding operating system, and also because it will take away the hassle of having to download any additional software to be able to run the application.

**Server Side:**

Heroku web services with Node.js will be used to implement the backend. The Free-Dev Plan from Heroku will provide 512MB of RAM power and a maximum of 1 web service, Node.js on the other hand will provide the web framework to connect the front-end and the database seamlessly. PostgreSQL will be use for the Database Management System (DBMS) and it is included in the package from Heroku. This web application will be manage with a GitHub account that will let developers push updates to their code and maintain multiple version of it available, in case any harm or corruption is presented in the latest installments. The back end will manage a payment system that will be used to collect funds from sponsoring companies to research groups and projects being developed by students at the university.

**Client App:**

The Application will act as an online portfolio to connect companies and possible future employers to students. Here, the students will be able to provide information such as personal projects, resume, contact info for the recruiters. They will also have the capability of looking for career fair related events and add them to a list. The recruiters will have the opportunity to search for students, and filter them out by major and gpa. In addition they will be able to search research projects and nominate them for possible sponsorship with their respective companies and they would have the option of providing funds through the application.

* 1. Students
     1. Will have their portfolio online to display their projects, resume, information, and picture
     2. Can access information about events and mark them to be reminder
  2. Recruiters (Companies and Graduate Schools)
     1. Will have the option to select and save the information of the students
     2. Can nominate research groups in the universities for sponsorship
     3. Recruiters will also have the opportunity to promote their on-campus events.

**Division of Labor:**

**Fernando:**

* Student Profile
* Sign Up Form
* Sign in From

**Wilfredo:**

* Main Page
* Transactions
* Recruiter Profile

**Luis:**

* Admin Profiles
* Password Recovery
* Research and Projects profiles

**E-R Diagram**

ER Diagram Phase2.png

**Table Schema**

***Main Tables***

**Projects**

CREATE TABLE public.projects

(

projectid integer NOT NULL DEFAULT nextval('projects\_id'::regclass),

title character varying(30) NOT NULL,

description character varying(30) NOT NULL,

student\_id integer,

image character varying(150),

CONSTRAINT projects\_pkey PRIMARY KEY (projectid),

CONSTRAINT projects\_student\_id\_fkey FOREIGN KEY (student\_id)

REFERENCES public.student (studentid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.projects

OWNER TO xwozcfrzmmekkv;

**Student**

CREATE TABLE public.student

(

studentid integer NOT NULL DEFAULT nextval('student\_id'::regclass),

name character varying(30),

lastname character varying(30),

email character varying(30),

gpa real,

year integer,

userid integer NOT NULL,

majorid integer NOT NULL,

CONSTRAINT student\_pkey PRIMARY KEY (studentid),

CONSTRAINT student\_majorid\_fkey FOREIGN KEY (majorid)

REFERENCES public.major (majorid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT student\_userid\_fkey FOREIGN KEY (userid)

REFERENCES public.users (userid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

**Users**

CREATE TABLE public.users

(

userid smallint NOT NULL DEFAULT nextval('user\_id'::regclass),

username character varying(30),

password character varying(30),

role character varying(30),

CONSTRAINT users\_pkey PRIMARY KEY (userid)

)

**Major**

CREATE TABLE public.major

(

majorid integer NOT NULL DEFAULT nextval('major\_seq'::regclass),

value character varying(20),

CONSTRAINT major\_pkey PRIMARY KEY (majorid)

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.major

OWNER TO xwozcfrzmmekkv;

**Research**

CREATE TABLE public.research

(

rid integer NOT NULL DEFAULT nextval('research\_id'::regclass),

title character varying(100),

description character varying(200),

professorid integer,

funding character varying(100),

CONSTRAINT research\_pkey PRIMARY KEY (rid),

CONSTRAINT research\_professorid\_fkey FOREIGN KEY (professorid)

REFERENCES public.professor (professorid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

**Professor**

CREATE TABLE public.professor

(

professorid integer NOT NULL DEFAULT nextval('professor\_id'::regclass),

name character varying(30),

lastname character varying(30),

email character varying(30),

userid integer NOT NULL,

CONSTRAINT professor\_pkey PRIMARY KEY (professorid),

CONSTRAINT professor\_userid\_fkey FOREIGN KEY (userid)

REFERENCES public.users (userid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.professor

OWNER TO xwozcfrzmmekkv;

**Recruiter**

CREATE TABLE public.recruiter

(

recruiterid integer NOT NULL DEFAULT nextval('recruiter\_id'::regclass),

name character varying(30),

lastname character varying(30),

email character varying(30),

company character varying(30),

userid integer NOT NULL,

CONSTRAINT recruiter\_pkey PRIMARY KEY (recruiterid),

CONSTRAINT recruiter\_userid\_fkey FOREIGN KEY (userid)

REFERENCES public.users (userid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

**Event**

CREATE TABLE public.event

(

eventid integer NOT NULL DEFAULT nextval('event\_id'::regclass),

title character varying(50),

description character varying(200),

date date,

recruiterid integer NOT NULL,

CONSTRAINT event\_pkey PRIMARY KEY (eventid),

CONSTRAINT event\_recruiterid\_fkey FOREIGN KEY (recruiterid)

REFERENCES public.recruiter (recruiterid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.event

OWNER TO xwozcfrzmmekkv;

**Files**

CREATE TABLE public.files

(

fileid integer NOT NULL DEFAULT nextval('files\_id'::regclass),

resumecv character varying(30),

profilepic character varying(30),

userid integer NOT NULL,

CONSTRAINT files\_pkey PRIMARY KEY (fileid),

CONSTRAINT files\_userid\_fkey FOREIGN KEY (userid)

REFERENCES public.users (userid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

WITH (

OIDS=FALSE

);

ALTER TABLE public.files

OWNER TO xwozcfrzmmekkv;

***Relations Tables***

**isGoing**

CREATE TABLE public.isgoing

(

studentid integer NOT NULL,

eventid integer NOT NULL,

status character varying(30),

CONSTRAINT isgoing\_eventid\_fkey FOREIGN KEY (eventid)

REFERENCES public.event (eventid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT isgoing\_studentid\_fkey FOREIGN KEY (studentid)

REFERENCES public.student (studentid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

**Recruiterfolder**

CREATE TABLE public.recruiterfolder

(

entryid integer NOT NULL DEFAULT nextval('folder\_id'::regclass),

recruiterid integer NOT NULL,

studentid integer NOT NULL,

CONSTRAINT recruiterfolder\_pkey PRIMARY KEY (entryid),

CONSTRAINT recruiterfolder\_recruiterid\_fkey FOREIGN KEY (recruiterid)

REFERENCES public.recruiter (recruiterid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT recruiterfolder\_studentid\_fkey FOREIGN KEY (studentid)

REFERENCES public.student (studentid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

)

**Research\_student\_rel**

CREATE TABLE public.research\_student\_rel

(

rid integer NOT NULL,

studentid integer NOT NULL,

student\_role character varying(30),

professorid integer,

CONSTRAINT research\_student\_rel\_pkey PRIMARY KEY (rid, studentid),

CONSTRAINT research\_student\_rel\_professorid\_fkey FOREIGN KEY (professorid)

REFERENCES public.professor (professorid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT research\_student\_rel\_rid\_fkey FOREIGN KEY (rid)

REFERENCES public.research (rid) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE CASCADE,

CONSTRAINT research\_student\_rel\_studentid\_fkey FOREIGN KEY (studentid)

REFERENCES public.student (studentid) MATCH SIMPLE

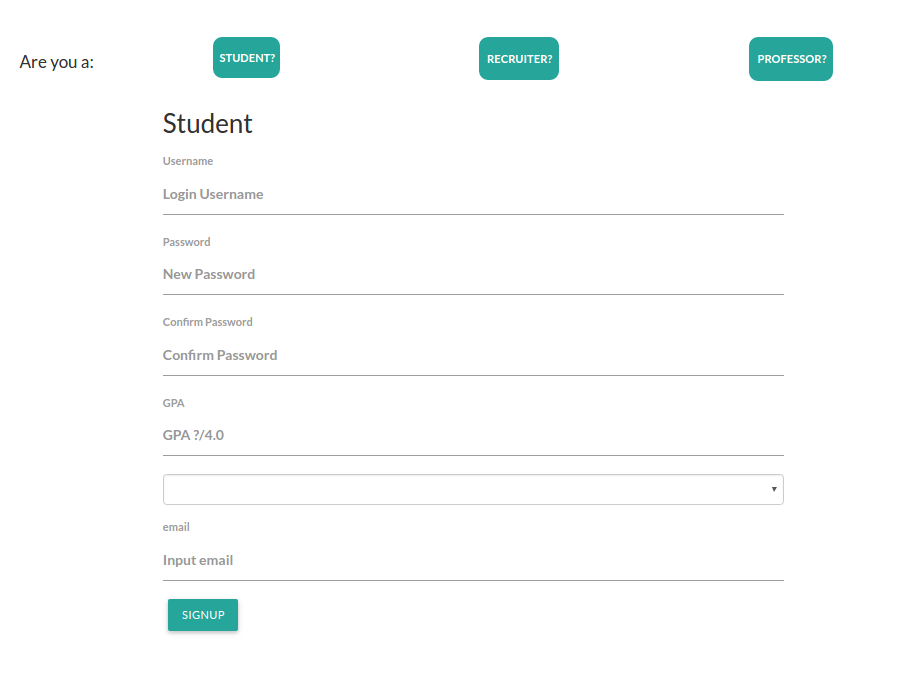
ON UPDATE NO ACTION ON DELETE NO ACTION

)

**ClientSide-Screen Shots**



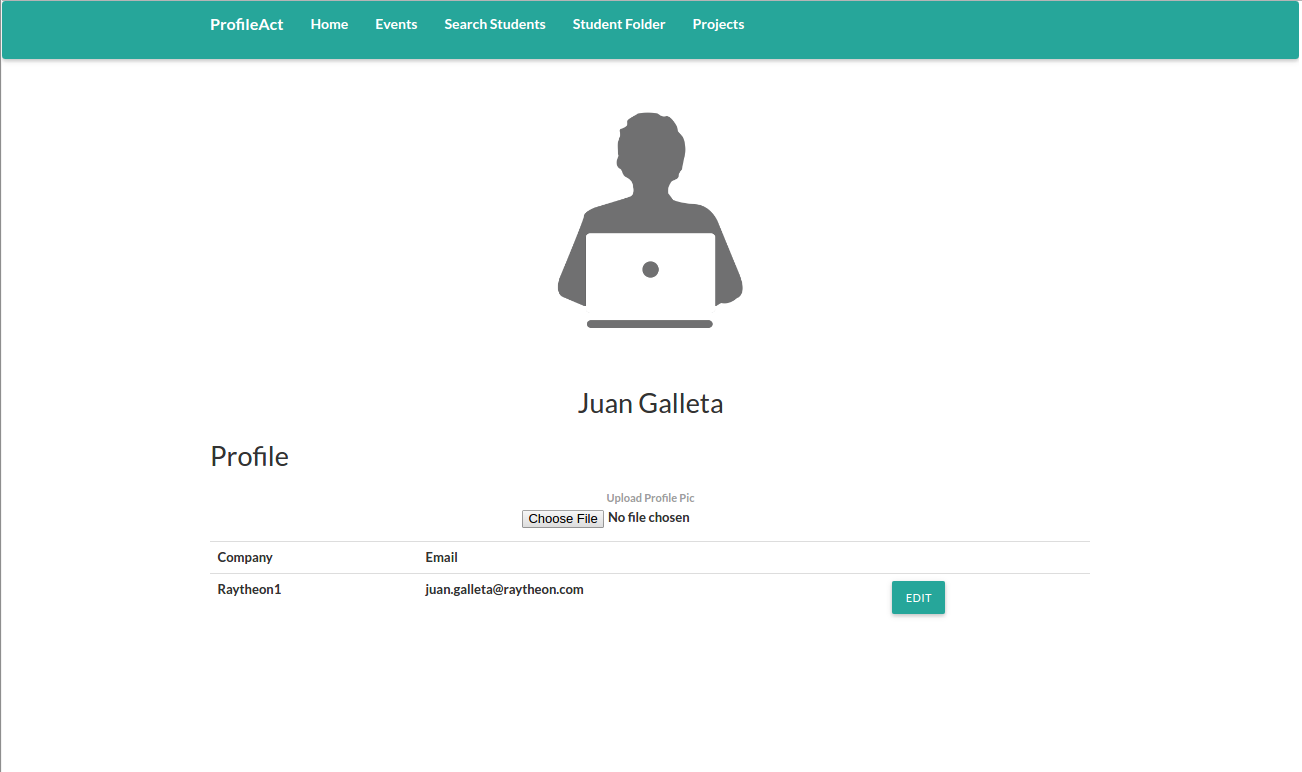
Launch Page



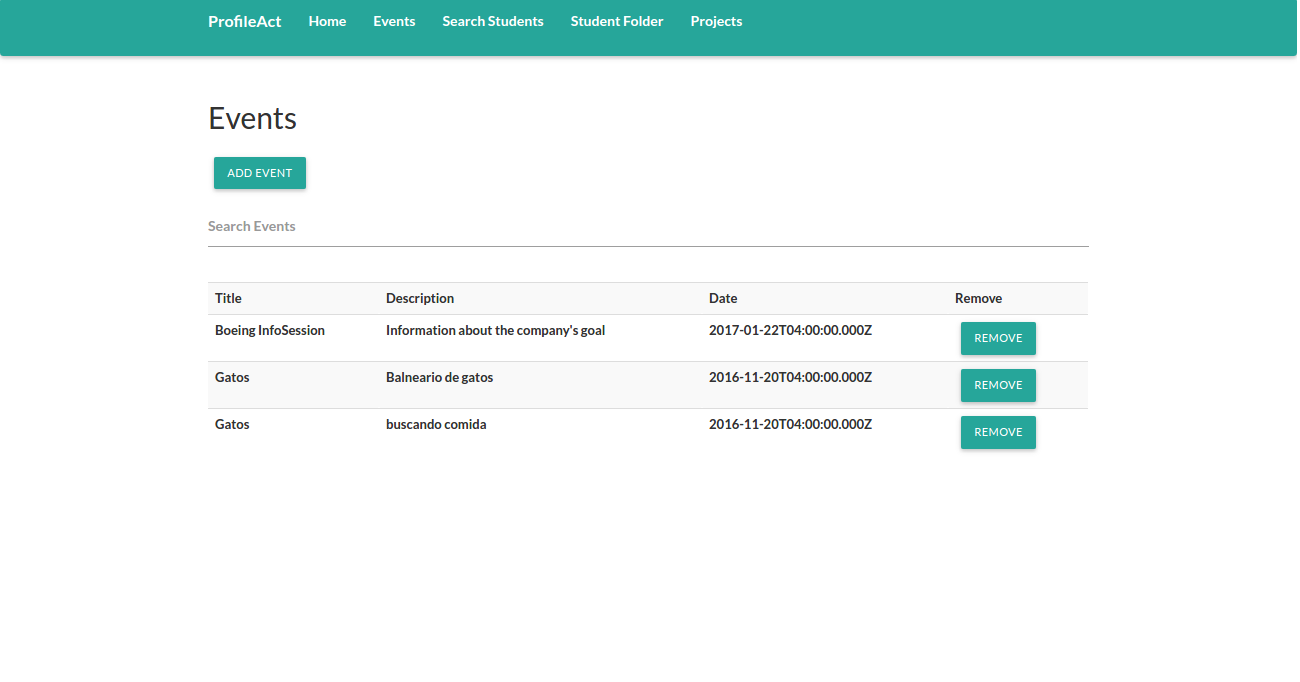
Sign Up



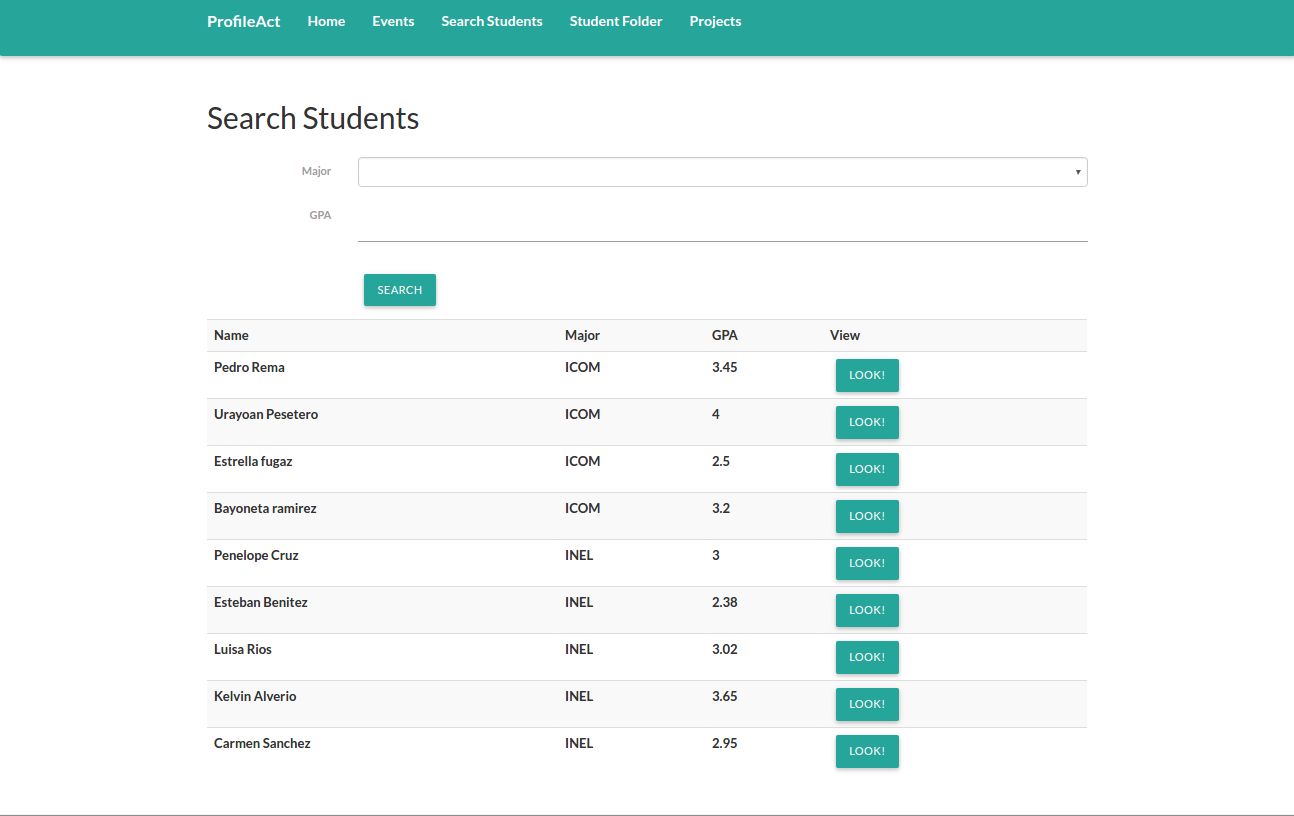
Sign In



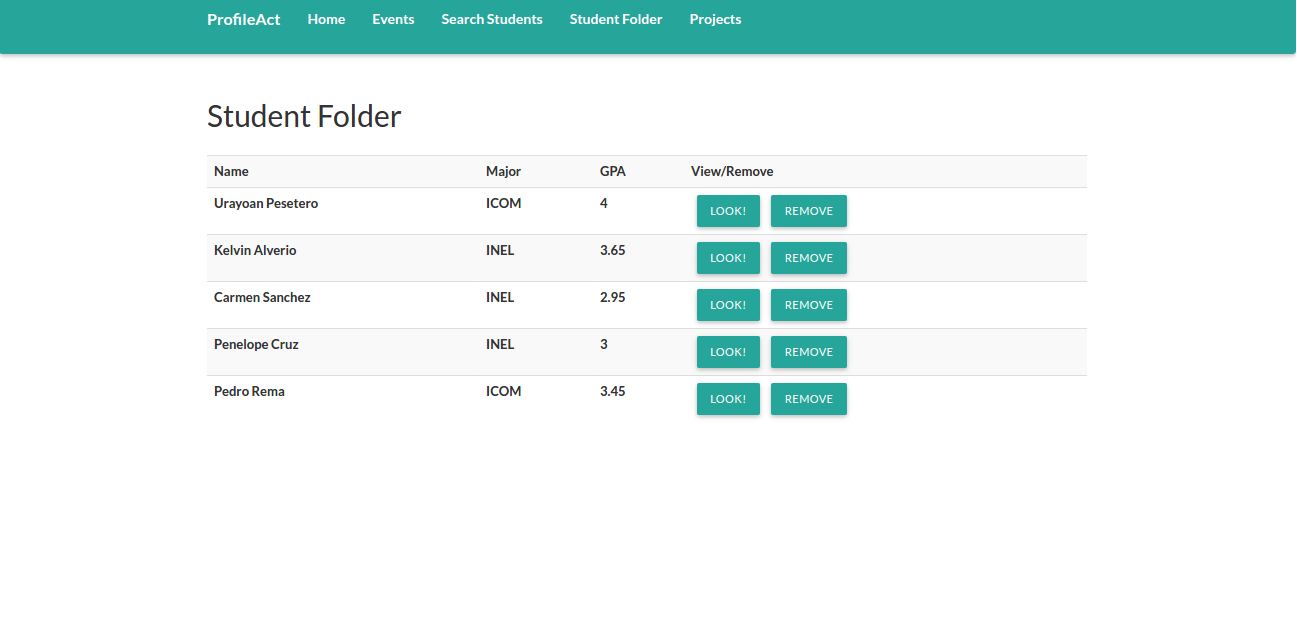
Recruiter Home Page



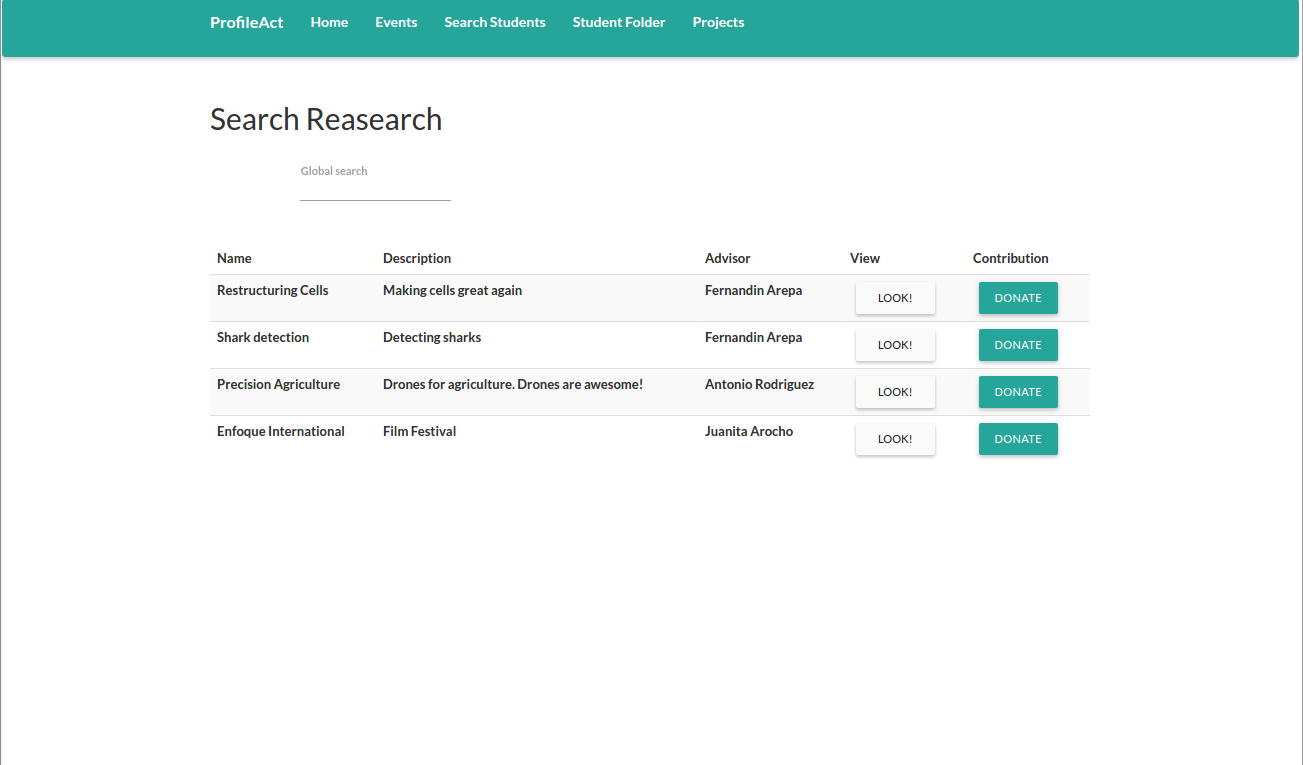
Recruiter Events



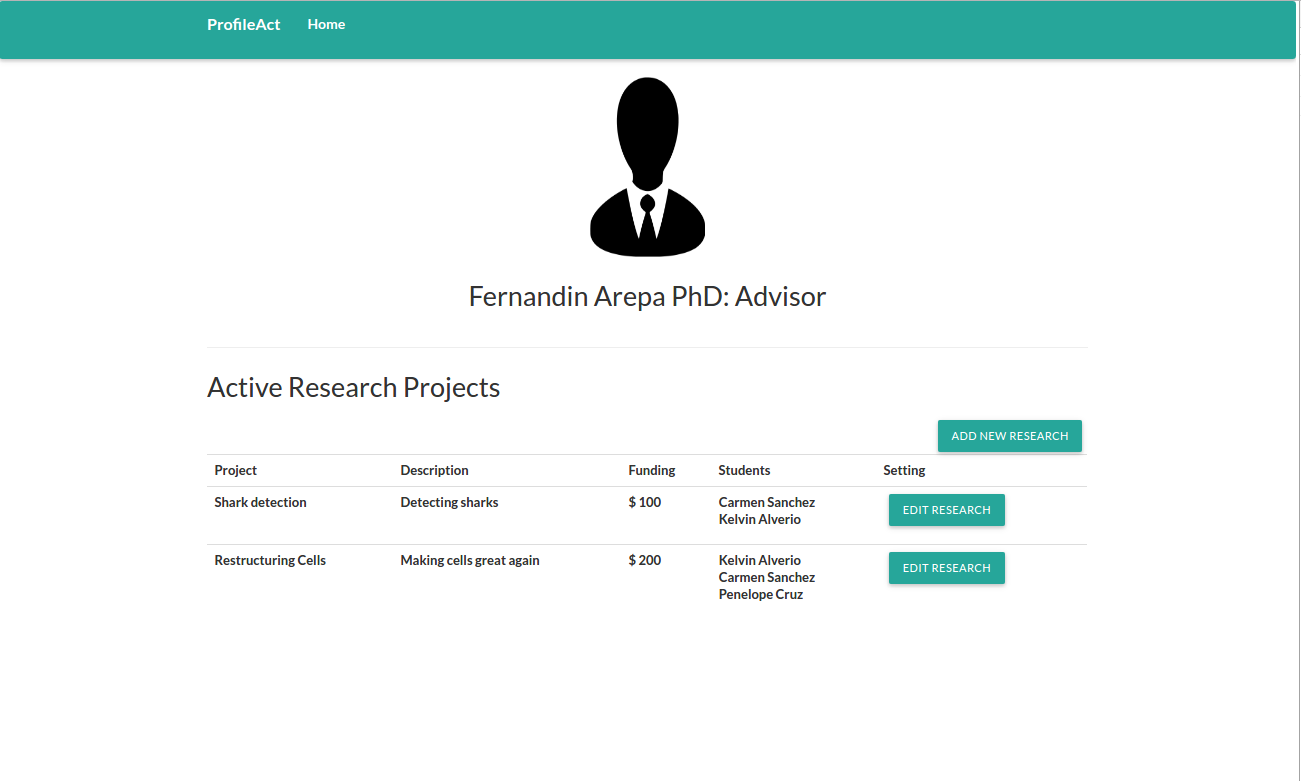
Recruiter Student Search



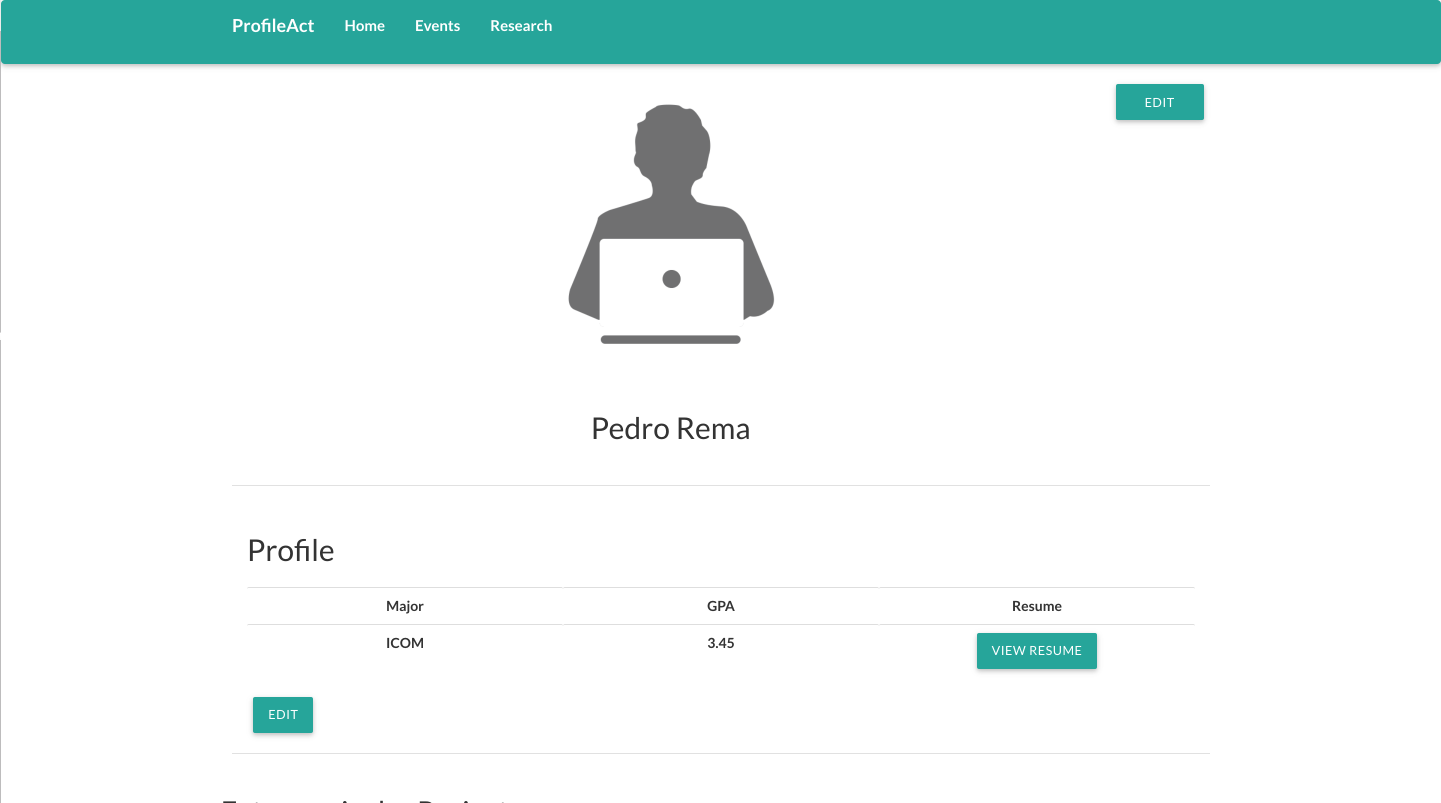
Recruiter Student Folder



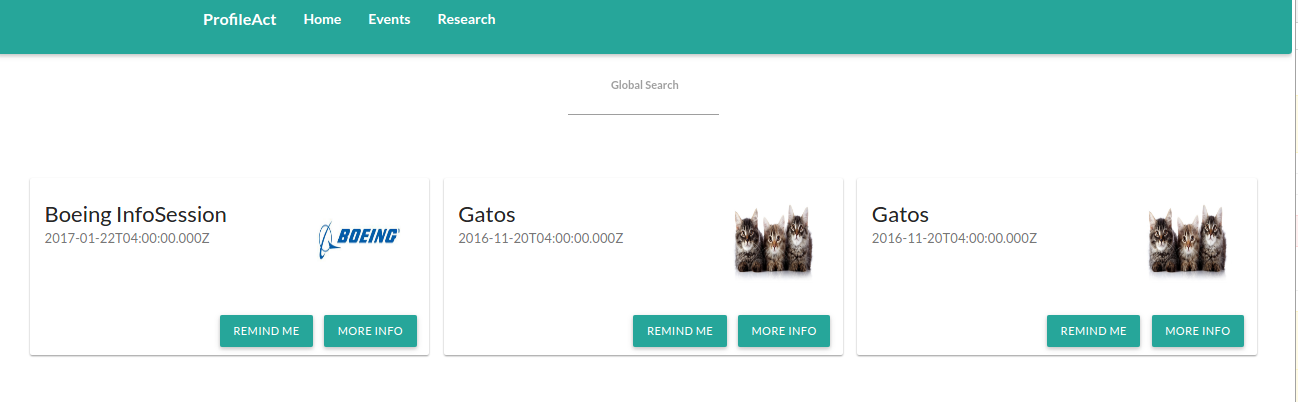
Recruiter View Research



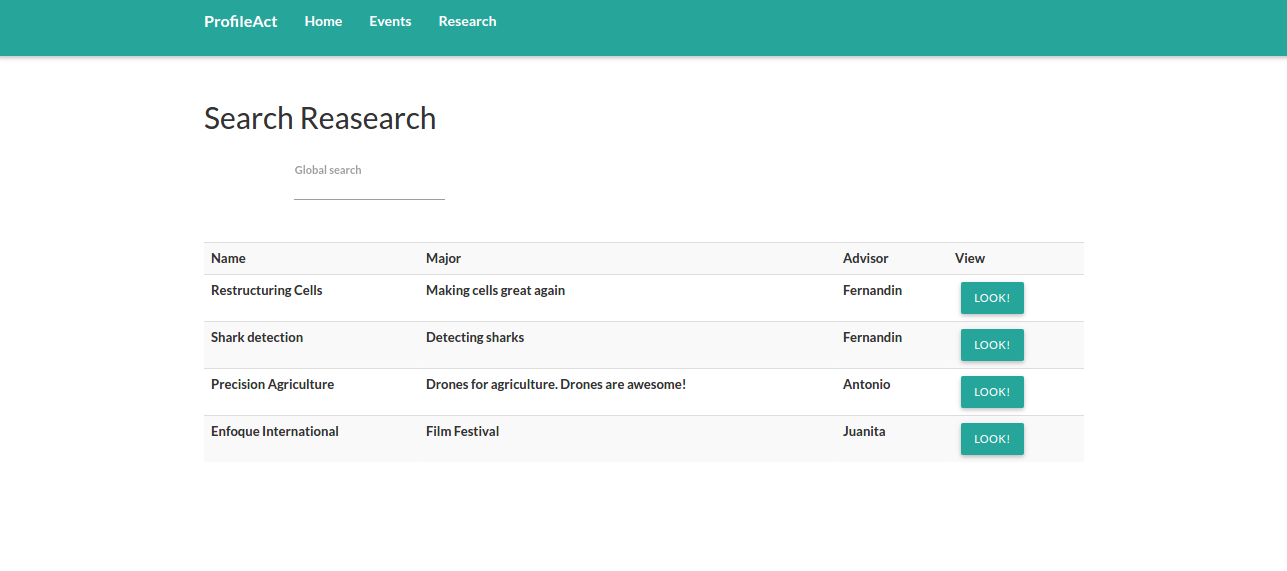
Professor Home Page



Student Home Page



Student Events



Student Research

Launch Page: First page you land when you go open the application. It provides with login and sign up forms.

Log In: Form to login

Sign Up: Form to sign up as one of the three users the application has

Professor Home Page: It provides with the information of the professor, he/she can edit his information and add researchs

Recruiter Home Page: It provides with the information of the recruiter, he/she can edit his information

Recruiter Events: Recruiters can create or delete events

Recruiter Search: Recruiters can search for students and view their information

Recruiter Search Research: They can view researches and their information and decide wheter they want to fund the research

Student Home Page: It provides with the information of the student, he/she can edit his information

Student Events: Students can view events and mark them to get reminders

Student Research Page: Students can view researches and their description