CS252: Final Project Report
Resume, Homepage creator

Ajay Sharma (12055) Arpit Shrivastava (12161) Varshini Kumavath (12372)
Pranav Maneriker (12497) Sakshi Gupta (12604)
Group 02

November 27, 2014

Abstract

The aim of this project is to create a resume and homepage building utility (as a aweb application). It allows the usage of multiple templates and user created templates as well. The software can be used for generation of resumes in various standard formats such as latex, pdf, and html. It also provides data in a json format for use in any other standard tools.

1 Problem Statement

The application must allow :

- Create an account for resume/homepage editing and saving multiple versions of resumes/homepages
- Use dynamic forms to input the data in some standard fields for the resume
- Allow people to use templates from a set of standard templates.
- Compare multiple versions of resumes
- Generating resumes in different formats (LaTeX, pdf, HTML etc.)
- Enable users to create and share customized templates
- support for LaTeX, MathML and other equation markup directly in the HTML source with no special browser setup required

2 Method Used

List of frameworks and tools used:

- Ruby on Rails
- Javascript, jQuery
- Database engine: SQLite for development and Postgres for production
- Pandoc, PandocRuby API
- wickedPDFRuby gem, wkhtmltopdf CLI tool.

- bootstrap-sass
- mathJax
- bcrypt-ruby
- will-paginate
- paperlip gem

2.1 Backend

Ruby on Rails MVC framework has been used to create the backend. The structure is as follows:

2.1.1 Database schema

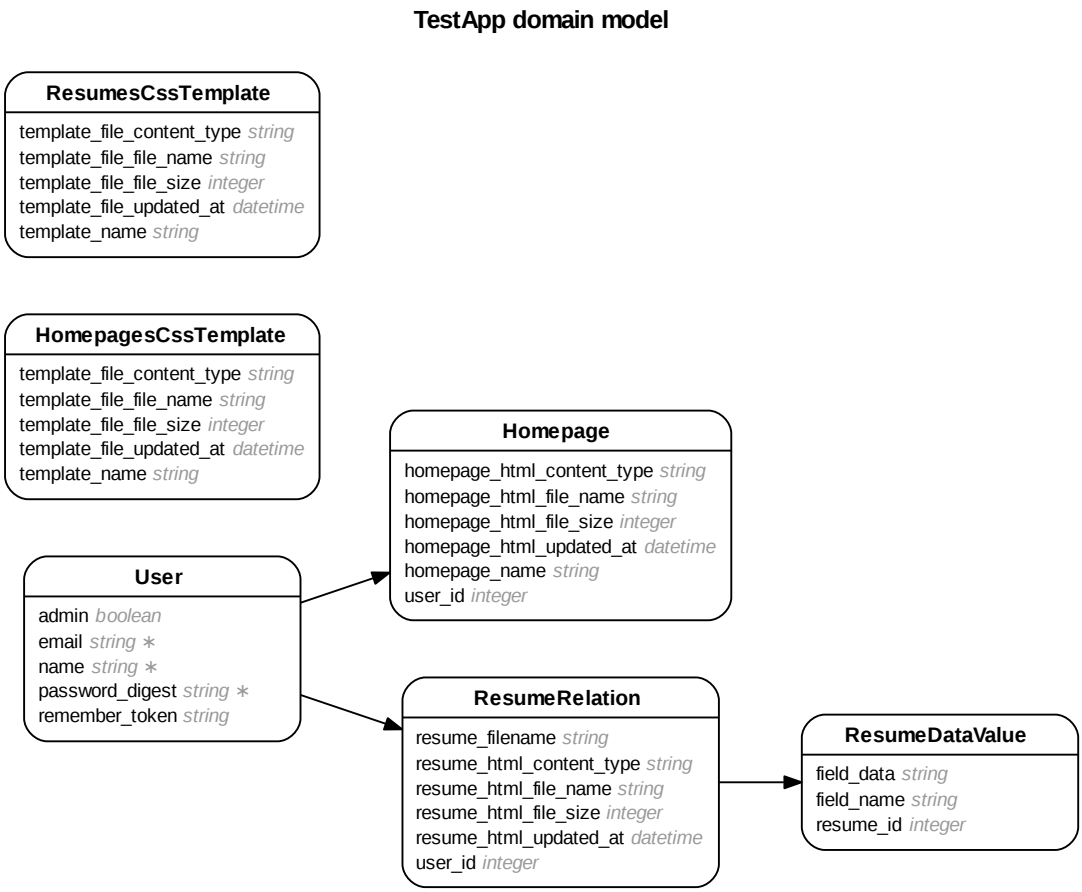

Figure 1: Entity Relationship diagram

2.1.2 Registration and login

bcrypt-ruby is used to generate encrypted passwords at the time of user registrations. The user specified email address is used as the user name and their avatar is loaded from gravatar.com.

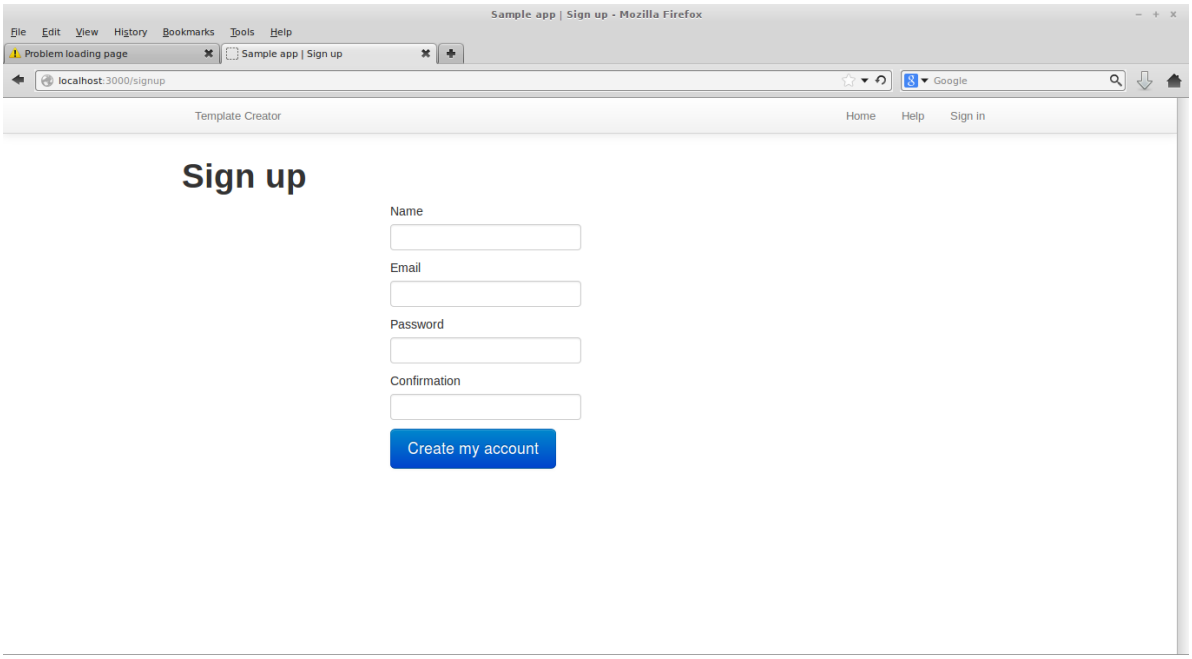

Figure 2: Signup Page

2.2 Resumes

2.2.1 Create Resume

We have used javascript for the dynamic resume/homepage updation. Every time a head/entry is checked/unchecked, function updateResume() is called to generate the modified html of resume/homepage and is rendered. Different templates have been incorporated by having different CSS files. When a particular template is chosen, the css is used for styling the resume.
For the input of resume data, user friendly form has been created which enables the hiding / displaying of particular head details by toggling the slider button. Similarly, checking the boxes adjacent to a field puts it on the resume/homepage and unchecking skips adding it. Checkflags have been used for each field to implement this. For the education section, a tabular entry is created for each of the class entered and checked. For the heads academic, extra-curricular achievements, work experience, papers published, a list of items is created by adding fields using the add button. Google font APIs have also been used to add some classic font styling. Different entities to be styled in the HTML have been given different classes for giving more control on styling. For instance, background colour, font colour, size, orientation, head background strips, tabular, list formats etc. In homepage, for navigation using the top menu bar, div ids of sections have been used and links to these have been added to the menu items.

Listing 1: jQuery code for dynamic form (jQuery)

```
function addAcadAchievements()
   {
     var data = "";
     data =data + '<div class="checkbox">\
```

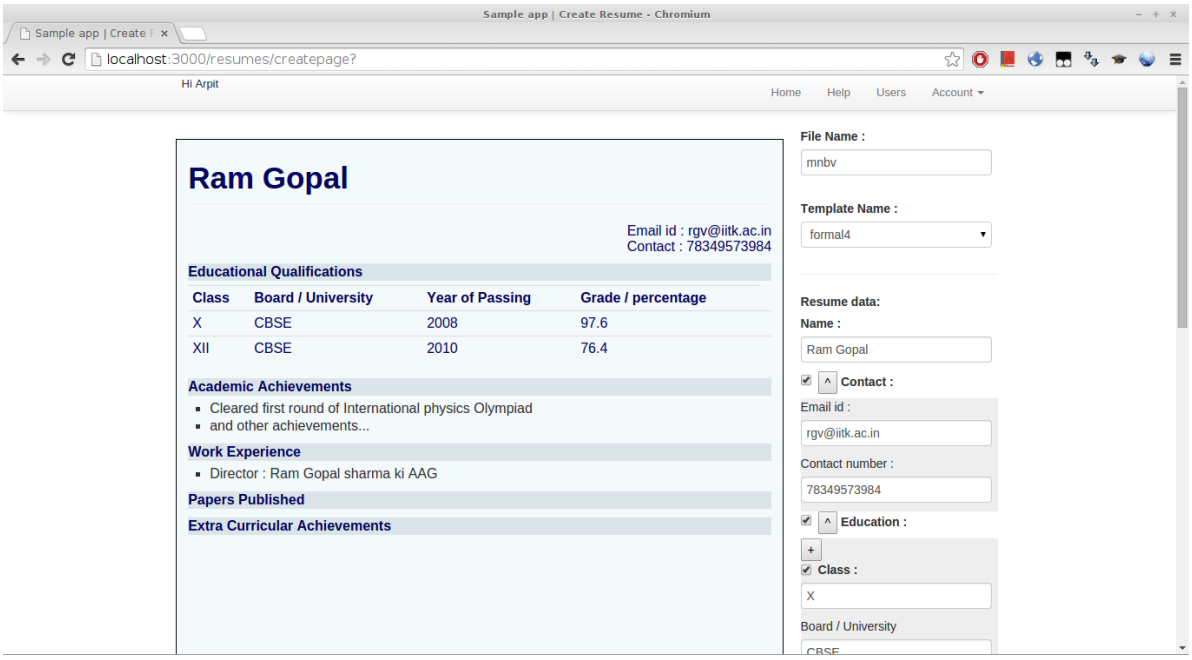

Figure 3: Create resume page

```
                <input type="checkbox" id="acadAchievcheck'+acadAchievements_index+'"
                    checked >\
            </div>\
            <input id="acadAchieve'+acadAchievements_index+'" type="text" name="
                acadAchieve'+acadAchievements_index+'" onchange="updateResume()">\
            <br>\
          </div>';
      acadAchievements_index += 1;
      $("#acadAchievementsgroup").append(data);
    }

function updateResume()
    {
      ..
   if(check("educationcheck"))
      {
        data = data + '<label class="section_title">Educational Qualifications</label><
            table class="table table-condensed">\
                  <tr>\
                   <th>Class</th>\
                   <th>Board / University</th>\
                   <th>Year of Passing</th>\
                   <th>Grade / percentage</th>\
                  </tr>'
        var i;
        for(i=0; i<edu_index; i++)
        {
          if(check("educheck"+i))
          {
            data = data + "<tr><td>"+getData("class"+i)+"</td><td>"+getData("board"+i)
```

```
                +"</td><td>"+getData("yearofpassing"+i)+"</td><td>"+getData("grade"+i)+
                "</td><td></tr>"
            }
          }
          data = data + "</table>";
        }

    }
...
```

2.2.2 Create Template

User friendly forms for creating new templates have also been provided. Users can modify the values of various fields of the stylesheets of the resumes and save the templates for later use. These templates can then be easily selected using a dropdown menu in the createResume page. For implementing the controls of this page the following jQuery plugins have been used:

- **Color picker:** jsColor library has been used.
- **Sliders:** jQuery UI plugin has been used. This is shown in the following code snippets:

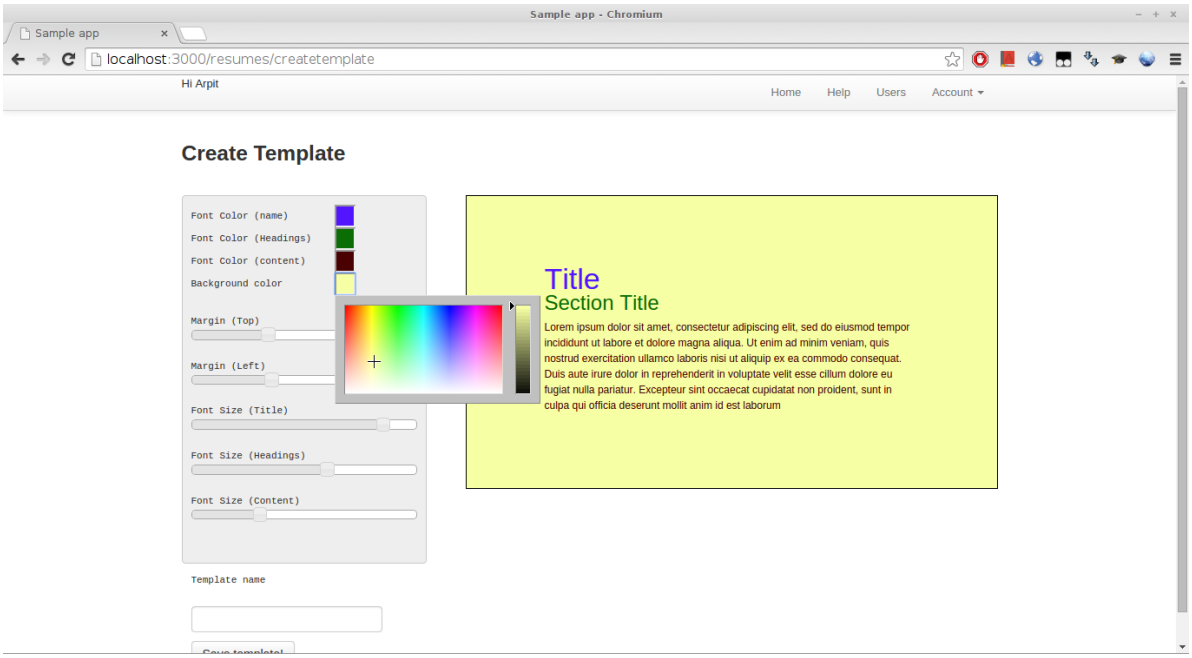

Figure 4: Create Template Page

Listing 2: Code to save the generated css file (Rails)

```
def postcssfile
    ActiveRecord::Base.transaction do
      @new_name = params[:file_name]
      @old_r = ResumesCssTemplate.find_by_template_name(@new_name)
```

```
      if !(@old_r.blank?) || (@new_name.blank?)
        flash[:error] = "You forcibly posted an invalid css file. Your css file will
            be lost"
        redirect_to root_url
      else
        file = StringIO.new('<style>' + params[:css_res] + '</style>')

        file.class.class_eval { attr_accessor :original_filename, :content_type }
        file.original_filename = @new_name + ".css"
        file.content_type = "text/css" # you could set this manually aswell if needed
             e.g 'application/pdf'
        ResumesCssTemplate.new(template_name:@new_name, template_file: file).save()
      end
      file.close
    end
  end
```

2.2.3 Existing Resume Operations

We show a list of all the resumes that were created. We also allow users to edit their existing resumes where the data gets pre-filled with the saved values before they edit the resume. Apart from that there are links for downloading the resume in various formats such as html, pdf, latex, and json.

Listing 3: Extra code to allow editing existing resume (Rails)

```
if !(session[:existing].nil?)
   @res_id = session[:resume_id];
   if !(current_user.resume_relations.find(@res_id).nil?)
   current_user.resume_relations.find(@res_id).destroy
   end
   session.delete(:existing)
   session.delete(:resume_id)
end
```

2.3 Homepage

2.3.1 Create Homepage

Dynamically created homepage with logic similar to that of resumes. The page also has internal links for scrolling etc.

2.3.2 Existing Homepage Operations

We show a list of all the homepages that were created. We also allow users to edit their existing homepages where the data gets pre-filled with the saved values before they edit the resume. Apart from that there are links for downloading the homepages.

2.4 Miscellaneous

We have used the javascript library mathJax to coompile mathML and Latex code given in form fields for display in the generated resumes and homepages.

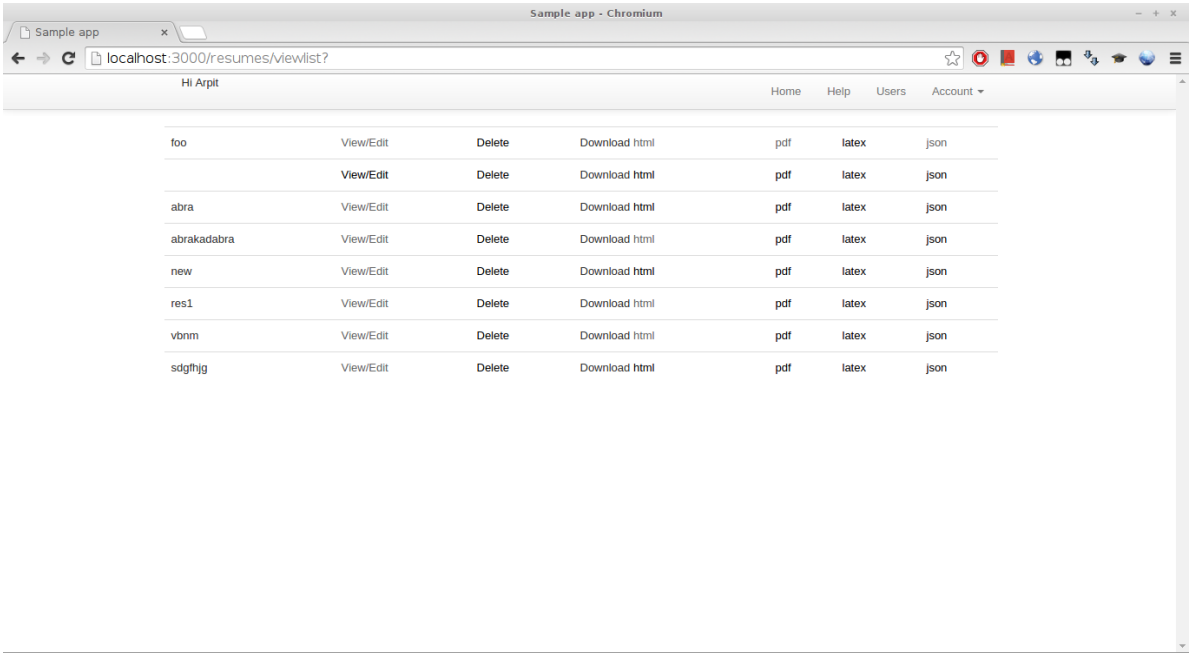

Figure 5: Existing Resume operations

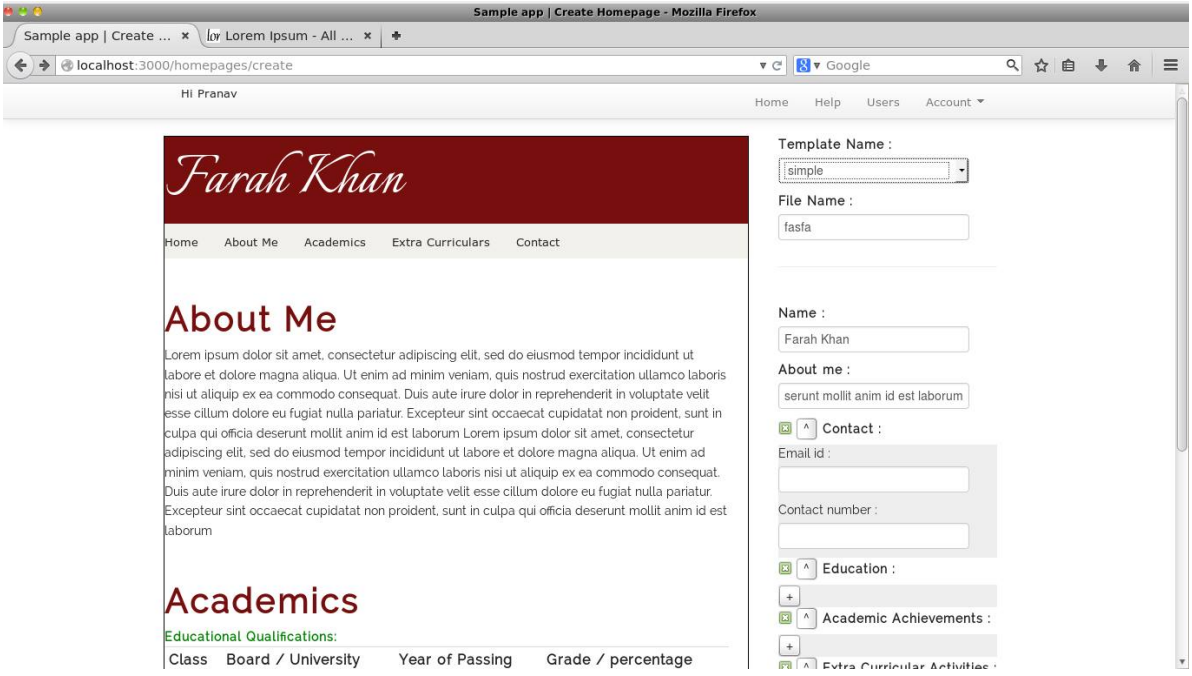

Figure 6: Create Homepage

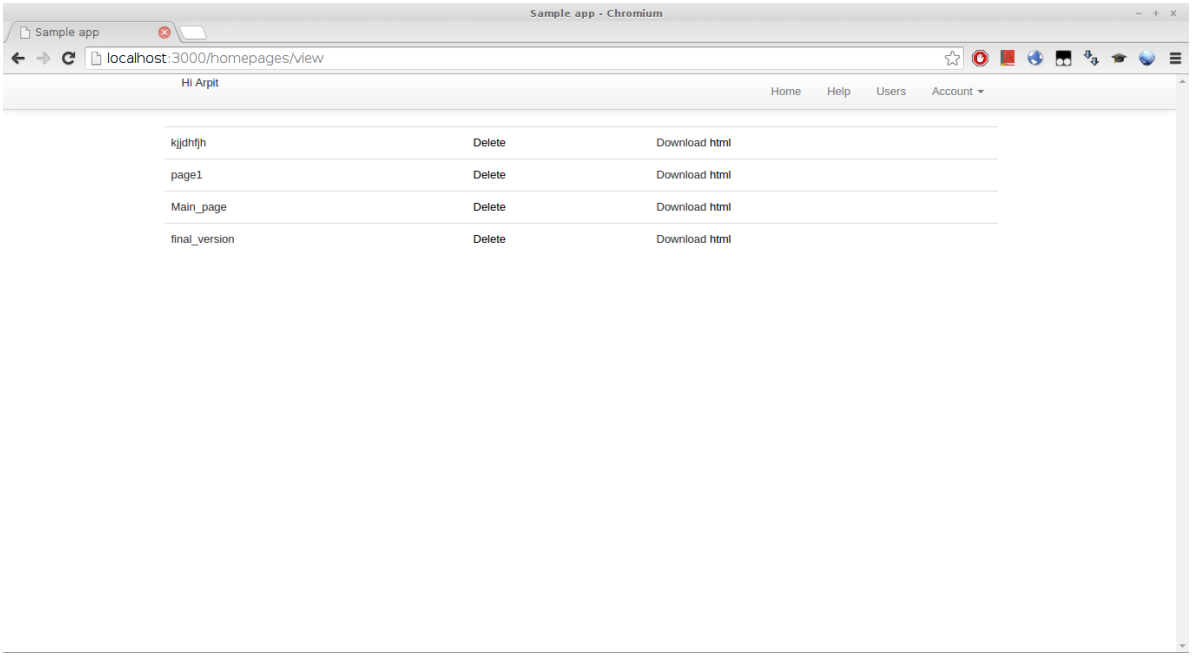

Figure 7: List of all the homepages created

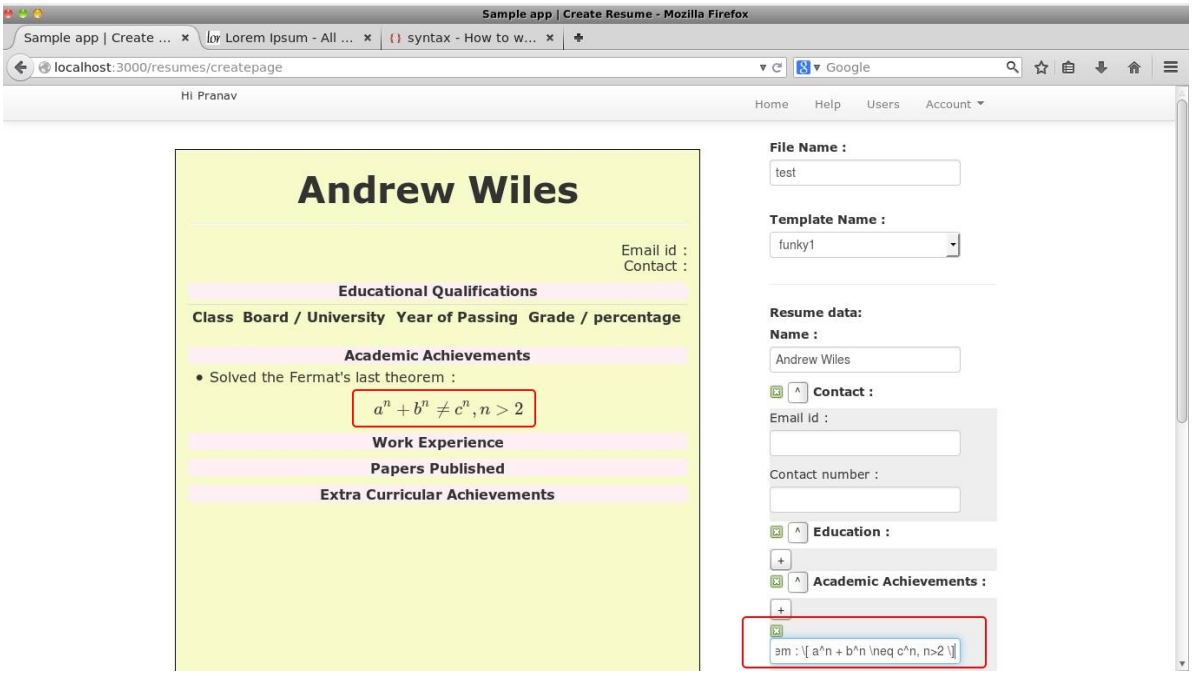

Figure 8: Figure showing MathJax example

3 Results and Conclusions

We have successfully created a dynamic resume/homepage generation application. Suppport for MathMl/Latex, various custom css templates are also available as ready to use tempates. A template created by any user can be used by any other user as well.
Future Work:

- Add an api to accept the json format and directly generate the resume. This will prevent inconsistency between varous resumes created.
- LaTeX rendering is not correct. Support for tables, equations is not good. This must be added
- Allow users to add fields to the resume. Some people may want to add custom fields such as books,

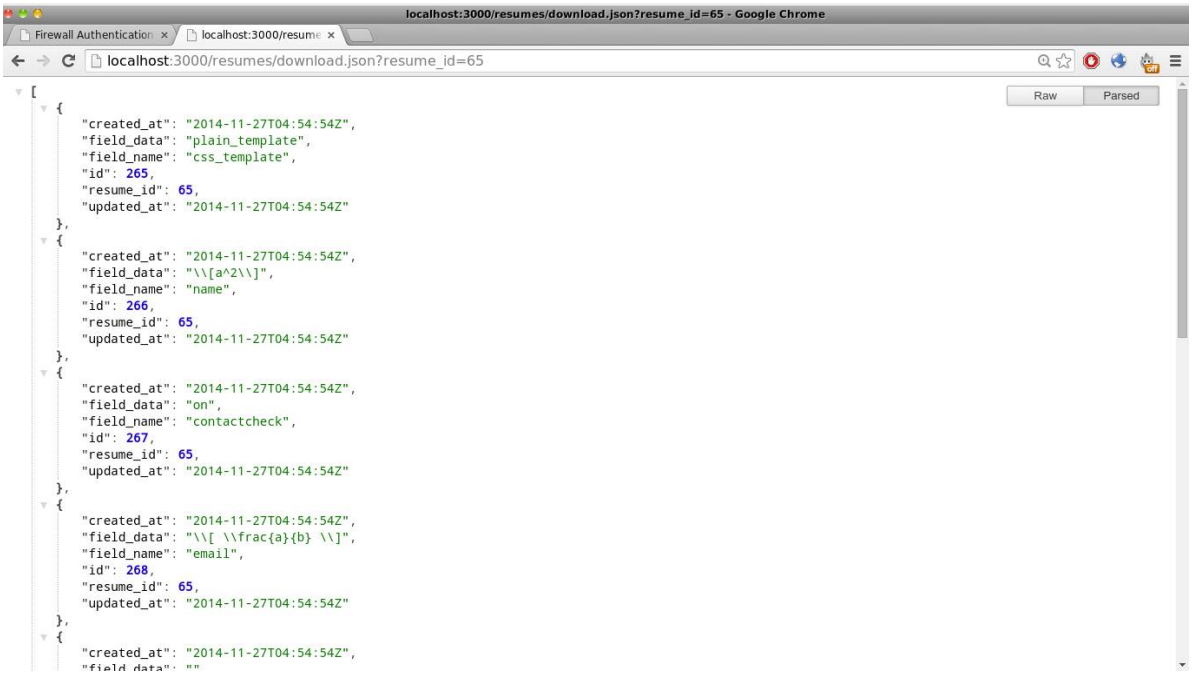

Figure 9: Figure showing the generated json

References

[1] Michael Hartl, *Ruby on Rails 3 Tutorial: Learn Rails by Example*. Addison Wesley, 3rd edition, 2010.

[2] Rolf Timmermans, *Rails-erd : Entity Relationship diagram generator*. 2014.

[3] American Mathematical Society (AMS) and the Society for Industrial and Applied Mathematics (SIAM), *MathJax*

[4] Jakob Truelsen and Ashish Kulkarni, *wkhtmltopdf*