Easy Rec

An escrow file server for letters of recommendation

Prepared by: Team EasyRec

Team Members:

Henry Eigen
Jovan Hernandez
Ria Patel
Ross Ketron

Team Name for GitHub Repository: EasyRec

INTRODUCTION

This project implements a web application that will allow a user easy access to recommendation letters from professors and colleagues. The motivation for this project comes from personal experiences amongst the team members. As we apply for various positions and internships, it is a necessity to ask professors and other professionals to write letters of recommendations for us. These letters might not seem very cumbersome, however, these professors receive numerous requests from students (sometimes one student may require several letters from a single source). This application aims to eliminate the redundancy and repetition associated with the reference portion of applications and provides a space where recommendation letters can be stored for future use.

During the development process, we encountered some changes in our project life cycle and our design. The development of a few application features were either delayed in our development plan or omitted from this final demonstration due to complexity and time constraints. Although it may seem that we had a lot of setbacks, we were able to reach our goals for a basic implementation of the application in time for this demonstration. If given more time to work on the project, we would definitely be able to have the full concept of the application completed.

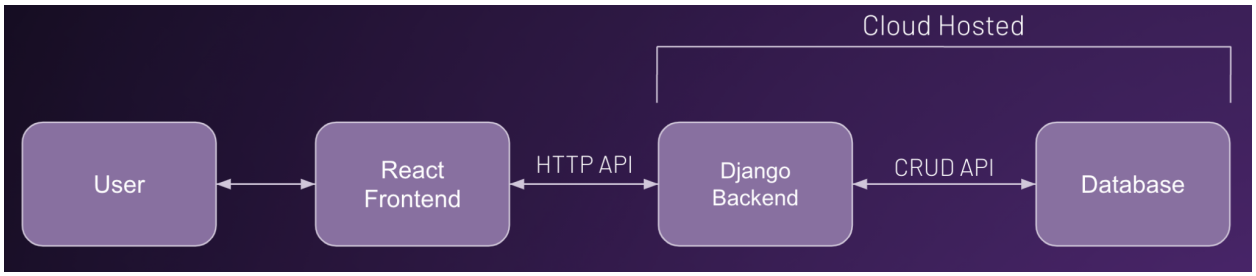
Customer Value

EasyRec did not encounter any changes for the Customer Value section since the submission of the project proposal.

Technology

EasyRec follows the system architecture shown in the high-level block diagram below:

Figure 1: System Architecture of EasyRec



The architecture changed somewhat throughout development. During development, we deployed the backend to the cloud separately from the frontend purely to ease development. We also stored all files and data in the SQL database instead of adding an independant file server. Also, we adjusted our original database schemas as we progressed to adapt to the requirements for both the backend and frontend. Finally, we adjusted the idea of having multiple user types and grouped them all together under one structure.

In order to ensure continuity of functionality during concurrent development across team members, we wrote a suite of tests to be run following each new. There are definitely some bugs in the app and it has some missing features that must be included if this were to ever become public. We tested our application by running a test suite on the backend API, however, due to time constraints, we were not able to run one for the frontend. The results are summarized in Figures 2 and 3.

Figure 2: Summarized Test Suit on Backend API

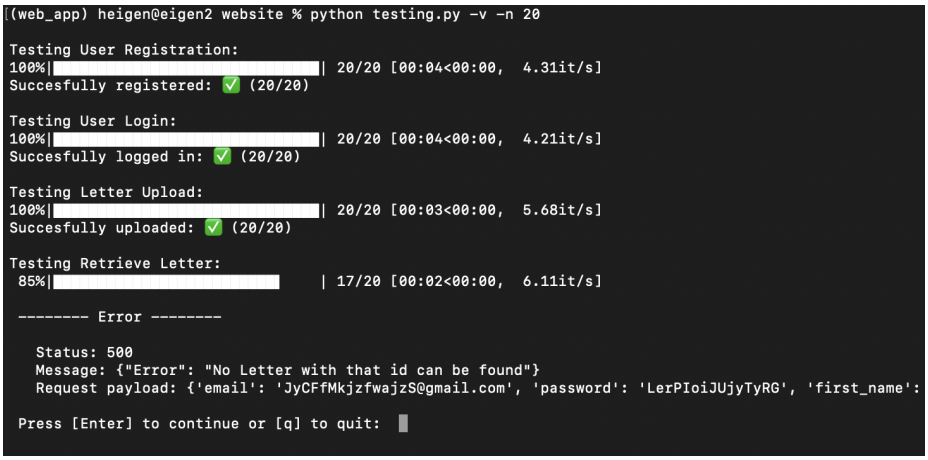


Figure 3: Detailed Test Suit for Backend API

Team

Our roles did not really change and everyone contributed wherever they could. Our roles are defined below:

Back-end Development

Henry Eigen

Roles: programmed back-end, designed schemas

Skills: Python, Django, Javascript

Ria Patel

Roles: Project Manager, assisted with Django framework, assisted with mock ups

Skills: Java, Javascript, Python

Front-end Development

Ross Ketron

Roles: Assisted with mock ups, collaborated on React components

Skills: HTML, CSS, Javascript, React

Jovan Hernandez

Roles: Assisted with mockups, collaborated on React components

Skills: HTML, CSS, JavaScript, React

Project Management

We completed most of our goals for the application in time for the demonstration, however, we were not able to fully execute our tentative schedule stated in the original project proposal. We had to delay some goals every sprint because some features took a lot of time to complete due to its complexity. As a result, we did not implement all of the features from our original design. These features include:

- Different user/account types.
- Authentication & Validation System
- Inbox requests for letters
- Referrers don't have ability to set permissions on letters they write or ability to edit letters (must re-upload).
- Human verification of created company accounts

Reflection

Project designing process went extremely well. We created mock ups for the frontend of the application and diagrams of the system architecture and they were a big help in the implementation process. Our development pipeline also worked really well as both teams worked concurrently. For example, as soon as a backend API piece was finished, it was immediately passed on to the frontend.

One thing that did not really go well was the backend API integration with the React frontend. There were many issues encountered during the process and they took a while to resolve. Also, the service to deploy the backend to the cloud didn't work well with the mixed framework apps, in our case, Django and React.