Fiscal Web Portal ENGINE Phase II

Students: XIAOTONG YANG, YUSIE YAO
INDUSTRY MENTORS: Ted Hanson, Bridget Faherty, Debbie Carnes, Shelley Prosise, Konrad Schroder, Al Brower, Jesse Chiem
FACULTY MENTORS: Payman Arabshahi
TEACHING ASSISTANT: Shruti Misra

Problem Statement/Objective

Currently, the College of Engineering lacks a tool to manage requests for administrative services, specifically financial transactions. Therefore, we were designed and developed a web application to serve as an advanced ticketing tool for UW College of Engineering departments to manage fiscal related tasks (reimbursement, travel, purchase, etc.) from users’ request submission through fiscal staff approvals and denial. The functionalities of this web application include:

• Submission of requests by users with the ability to route for appropriate approvals, upload documents, and get a receipt number.
• The budget approvers of that ticket approve or decline the request.
• Fiscal staffs review the approved requests again.
• System administrator manages units, subunits, budgets, and people.

Implementation

1. Submit a Request (Submitter)

   - Approve Request
   - Submit Request
   - My Requests

2. Approve / Decline a Request (Approver)

   - Approve Request
   - Submit Request
   - My Requests

3. See the Audit Table

   - Pacific Time Zone
   - Approval Request
   - Submission Request
   - My Requests

Requirements

- Understand the business goals
- Develop New Login Authentication & Authorization workflow: SSO into the application via UW NetID.
- Revise REST API calls
- Display submitted requests module
- Draft for notifications for approvers
- Complete the workflow
- Audit Trail Table implementation

Tools

Front-End Service:

We utilize the React and Redux, with extensive usage of Ant Design as React UI Components library and Redux Thunk as Middleware. This RESTful front-end service renders the page, interacts with the users’ actions, and sends different APIs (Application Programming Interface) provided by the back-end service. We also use create-react-app as an integrated toolchain to save the trouble dealing with module bundler and compiling the JavaScript, because under the hood of create-react-app, it uses Babel and webpack. Also, we handle authentication through react-google-login npm package.

Back-End Service:

We use Koa2 as the web framework for Node.js. We connect the backend with MongoDB Atlas Database, with Mongoose managing the relationships between data and providing schema validation. This RESTful backend-service contains queries to CRUD (create, read, update, delete) the database, responding to API requests sent by the front-end service.

Conclusion, References, and Acknowledgments

This application consists of 4 main levels:

- Requester layer to submit requests (6 forms).
- Approver layer to approve requests for specific budgets.
- Fiscal staff layer to deal with requests under within a unit.
- Administrator layer to handle the configuration of units, subunits, budgets, and people.

We refer to some official documents for different languages and frameworks:
- https://reactjs.org/docs/create-a-new-react-app.html (create-react-app)
- https://ant.design/components/overview/ (Ant Design React UI Components Library)
- https://www.npmjs.com/package/react-google-login (react-google-login)
- https://mongoosejs.com/docs/api (Mongoose)

DISCUSSION/FUTURE WORK

For the future work, development-wise, we can consider the following:

- We can provide a way for users to edit their requests after they are rejected by the approvers. We need to utilize JWT (JSON web token) for API Protections, limiting access to API calls. Also, we can let the user upload Excel sheet rather than input data for every budget. We can develop an email notification system to alert the approvers as well. Operation-wise, we need to create a pipeline to support continuous integration and delivery, and eventually deploy the service into the cloud platform.

This application was created under the supervision of:

ADVISOR: Ted Hanson
SPONSOR: College of Engineering