Capstone Project Proposal

GitHub Username: khargeprachi

Vendor Man

Description

A vendor management system is an internet based software application, which allows businesses to manage their permanent and temporary staff along with project management and payroll management. Various VMS exist such as Tipalti, Connecteam, Lystable and more.
Vendor Man is an application which basically focusses on the vendor engagement part of VMS and facilitates the communication between providers and suppliers and automates the staffing procedure along with managing contingent workforce. It provides an interface to view and manage the vendor registrations, their approvals, contracts and various project groups.

Intended User

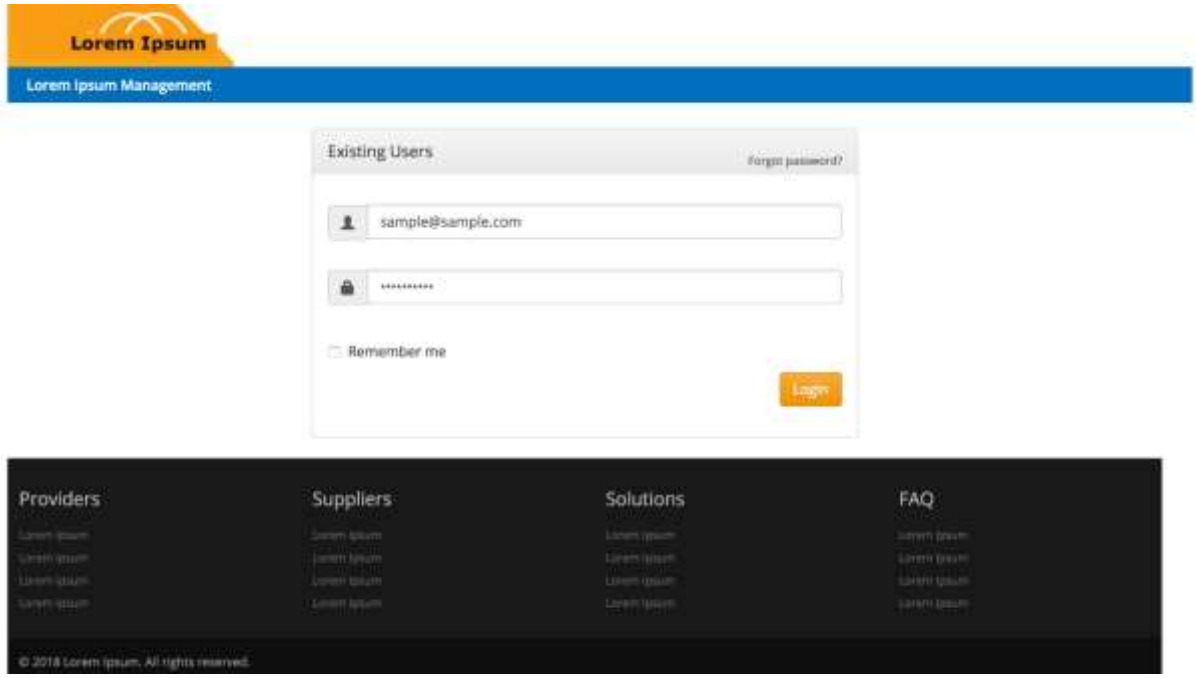
Any setup that has a contingent workforce involved can use this application. Companies and organisations using information technology as a base to manage its functioning, can utilise this application to keep track of different vendors, suppliers and service providers to ensure maximum productivity.

Features

Vendor registration: The up-to-date status of all the registered vendors is determined and can be an important factor in determining the over-all productivity. Vendor registered groups and sub groups can be used to ensure a proper balance in staffing.
Vendor approval: The current status of any vendor, approved or terminated, should be allowed to change by the authorities at any moment. This allows easy re-allocating and division of the workforce.
Contracts: Contracts can be used to decide the terms and conditions of the agreement. These can be used to test whether all mutually agreed conditions are satisfied or not. Contract extensions can take place any time.
Reports: Reports in a vendor management application should be regularly updated recording and monitoring the ongoing activities. These reports can be referred anytime for evaluation and any back tracking if required.
User authentication: Any user logging into the system must be authenticated to maintain confidentiality and avoid any leaking of information to an outside party. This application also helps vendors keep track of their engagements and companies keep track of their workforce.

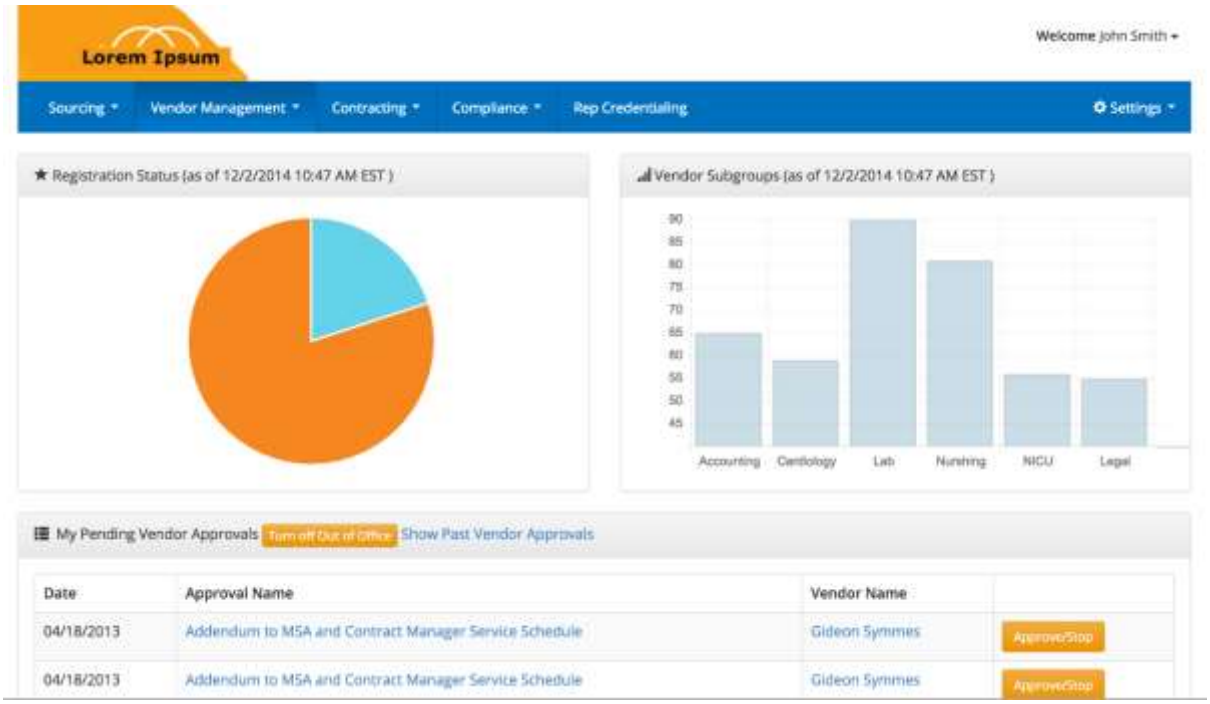
User Interface Mocks

Screen 1



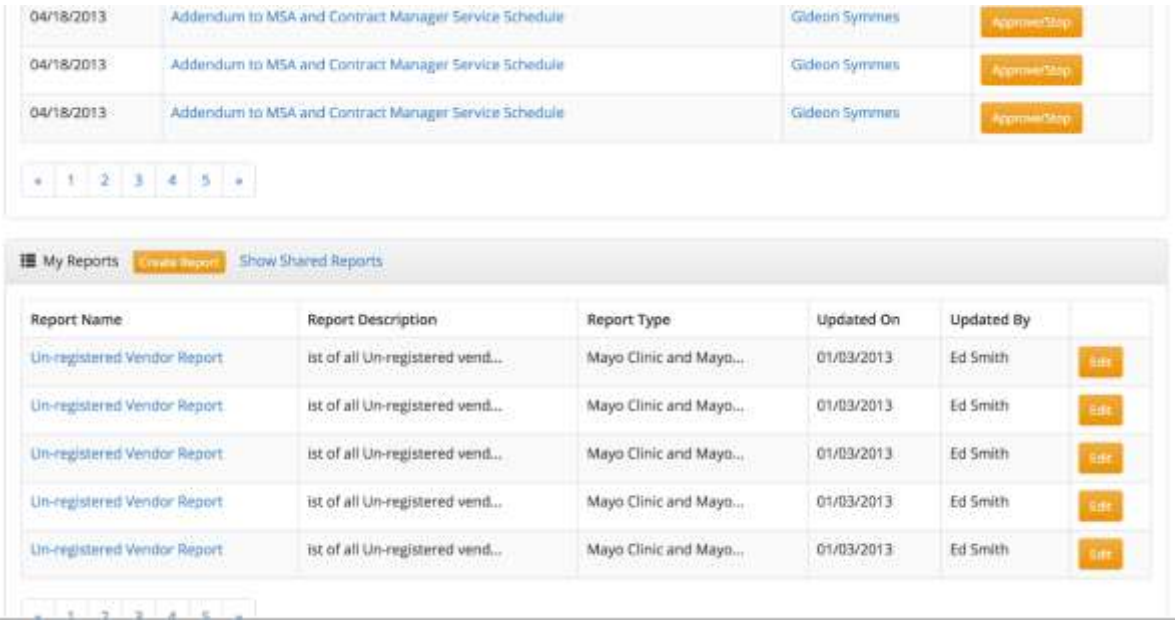
User authentication is done using a user id and a user password known only to the registered user. A forgot password option is also available which will reset the password after the user identity is verified.

Screen 2



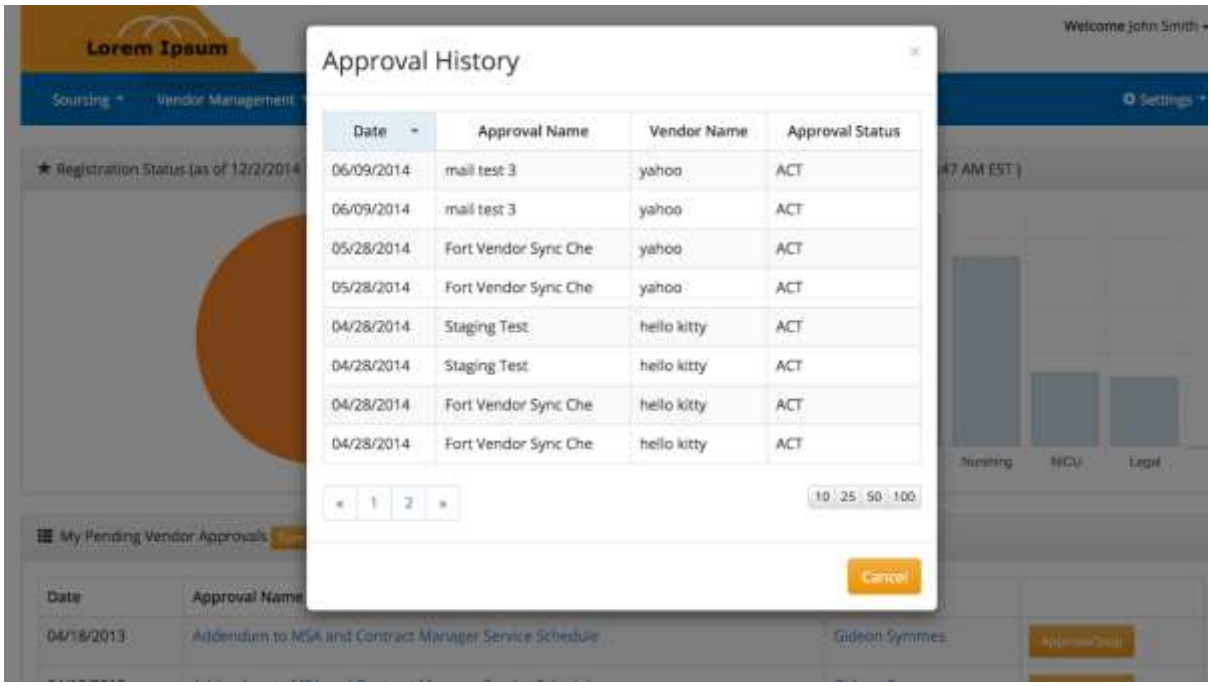
The up-to-date status of all the registered vendors is displayed using a pie diagram and can be an important factor in determining the overall productivity. Vendors are grouped and further divided into sub groups, the details of which, are shown using a bar diagram. Pending vendor approvals are listed with the date, approval name and vendor name along with a toggle option to approve/stop it.
A settings option is also provided so that user can edit its account information.

Screen 3



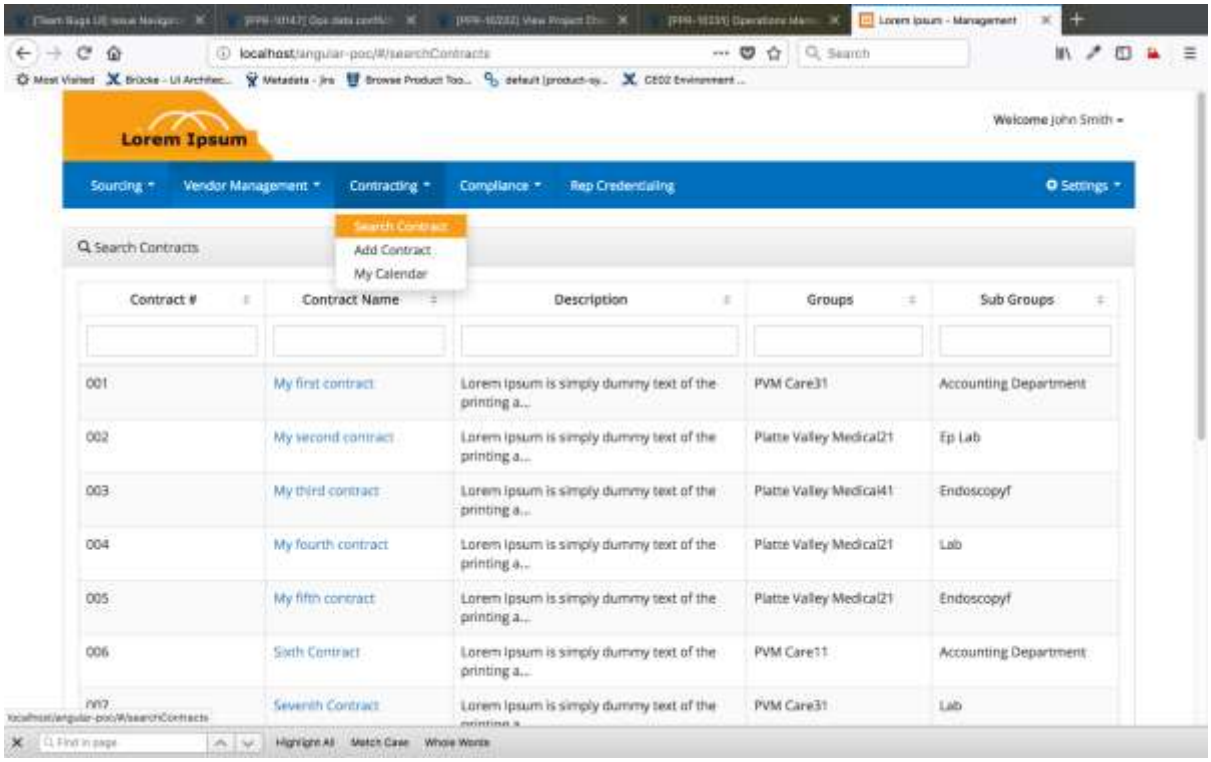
All the vendor reports are listed with details such as report name, report description, report type, and its last updated timestamp along with an updated by name. A button is displayed which allows the user to edit the corresponding report.

Screen 4



An approval history is made available to the user which shows the past vendor approvals.

Screen 5



In the contracting section, users can create a new contract by providing all the required information such as Contract name, its Description, groups and sub groups involved. A unique Contract id is generated by the system. The users can also see and search contracts. My calendar option is made available for scheduling purposes.

Key Considerations

How will your website handle data persistence?

Mock Json Server is used to handle data persistency. We create a fake REST API using json server package which stores all the data locally in a json file and any changes made are updated instantly. This helps us to focus and concentrate on the functioning of the applications without worrying about handling and supporting a full-fledged API.

Describe any edge or corner cases in the UX.

- On logging in successfully, the user is directed to the vendor management section.
- On hitting the logo at any instant, the user is again redirected to this section.
- Selecting the options in the navigation bar, redirects the user to corresponding section.
- While viewing Approvals history, the focus is locked inside the modal box, and user can return by clicking the 'Cancel' or 'Close' button.
- On logging out, the user returns to the main login page.

Describe any libraries you'll be using and share your reasoning for including them.
This project uses React Libraries to create the project due to its reusability of self-contained components and great developer tools. React uses a virtual DOM which allows it to minimize the amount of changes required between consecutive UI renders. It avoids the recalculation of all the UI properties for each and every render.
React follows one way data flow, i.e., only change in data can result in change in the UI. This makes sure that there is only one base and a single source of information, thus avoiding any collision.

Next Steps: Required Tasks

Task 1: Project Setup

- Install NodeJS and npm.
- Run npm install create-react-app. We can then use this react framework as a base for our project.
- Run npm install -g mock-json-server to set up a database server. This creates a json file which stores and updates data locally.
- Run npm start to execute in the development environment and npm run build for the production environment.

Task 2: Implement UI for Each Web Page and Frame

- Build UI for Login page. A forgot password option should also be added.
- Build UI for the vendor management section showing the percentage of registered workers, displaying all vendor subgroups and their counts in the form of a bar diagram and create an unordered list for showing the pending vendor approvals ordered by date of approval along with an option to open the approval history modal box. Also, insert a toggle button for each pending approval to approve/stop it.

- In the vendor management section, also showcase an unordered list, ordered by their time of latest update, of reports created by the current user along with any shared reports. An edit report option should also be available to the user.
- Build UI for the contracts section. All contracts that the current user has been involved in, should be listed. The user should also be able to search through the contracts, add a new contract and view or update his/her schedule on a calendar.
- A settings option is also available with its position fixed on the navigation bar. This should be a dropdown menu allowing user to edit its account information and other such details.

Task 3: Database Set up

- The details of all the registered users with their user ids and passwords are stored in a database and referred during user authentication.
- A database of all the vendors is created with their current registration status, group, sub group, approval names and contract ids. This database should be updated dynamically and can be referred to display all the information mentioned in the UI details.
- A separate database for all the contracts, reports, and approvals should be created with vendor ids in each record to match them accordingly and display information corresponding to a particular user.
- Searching contracts can be implemented using contracts database, approvals history can be viewed using approvals database and the user profile can be retrieved from the user database.

Task 4: React Components

- The application can be divided into different independent components such as app, approvals, contracts and reports.
- A main app component should update the default values of the state. On any change in the database, this state should update, thus reloading the application. In this way, the informative diagrams and lists will display the latest data.
- The main app component mounts the rest of the components only once the user is authenticated.
- Any asynchronous functionality should be included in the parent components only, as it ensures more usability of the components.