Expense Reimbursement System

# Executive Summary

The Expense Reimbursement System (ERS) will manage the process of reimbursing employees for expenses incurred while on company time. All employees in the company can login and submit requests for reimbursement and view their past tickets and pending requests. Finance managers can log in and view all reimbursement requests and past history for all employees in the company. Finance managers are authorized to approve and deny requests for expense reimbursement.

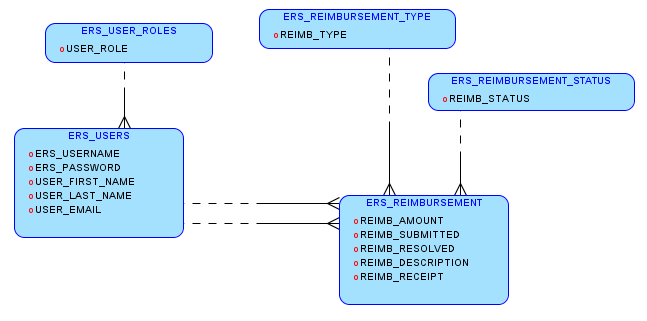
**State-chart Diagram (Reimbursement Statuses)**



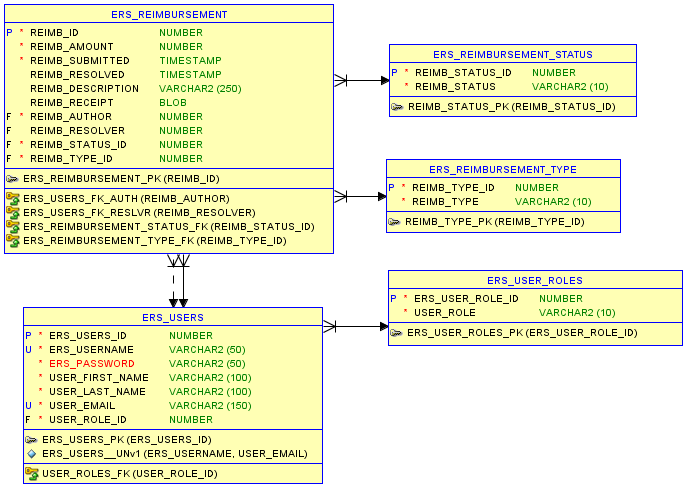
**Reimbursement Types**

Employees must select the type of reimbursement as: LODGING, TRAVEL, FOOD, or OTHER.

**Logical Model**



**Physical Model**



 New FK Relation : Used to create foreign key relationships between tables

#### **Things to note:**

##### **Dotted Lines means - Weak (Non-Identifying) Relationship**

* Entity's existence is independent of other entities
* Primary Key of child entity doesn’t contain Primary Key of parent entity

##### Solid Lines means - Strong (Identifying) Relationship

* Child entity's existence is dependent on parent
* Primary Key of child entity contains Primary Key of parent entity

**Use Case Diagram**



**Activity Diagram**



# Technical Requirements

The back-end system should use a NodeJS server to communicate with a PostgreSQL database. You may use pg, knux, or any appropriate ORM technology for communication. The NodeJS application must expose a RESTful API that meets the specifications for level 2 on the [Richardson Maturity Model](https://martinfowler.com/articles/richardsonMaturityModel.html) and follows general resource [naming conventions](https://restfulapi.net/resource-naming/). You must build a client-side application which consumes this API and offers all functionality described in the activity and use-case diagrams. The client-side application may be built using HTML/CSS/JS or using React. Additionally, you should include some client-side design library such as Bootstrap or Material-UI. User Experience will be a consideration for evaluation of this project.

## Testing Goals

* >80% test coverage for server
* >50% test coverage clientside

## Process Recommendations

* Commit AND push code daily
* Communicate blocking problems that impede progress promptly

## Optional stretch goals:

* Supervisor or auditor roles which can audit decisions made by finance managers.
  + Allow users with denied requests to appeal decisions to auditors.
* Enforce special requirements for very expensive reimbursements.
* More detailed/fine-tuned search functionality.
* Other appropriate features that suit the application