‘’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.

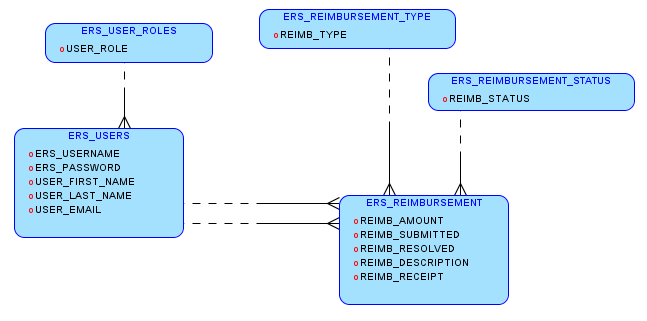
**State-chart Diagram (Reimbursement Statuses)**



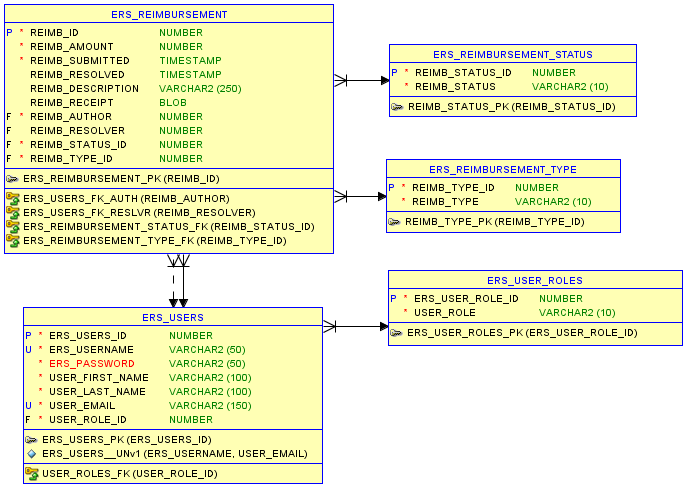
**Reimbursement Types**

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

**Logical Model**



**Physical Model**



**Use Case Diagram**



**Activity Diagram**



# Technical Requirements

The back-end system shall use JDBC to connect to an Oracle database. The application shall deploy onto Tomcat Server. The middle tier shall use Servlet technology for dynamic Web application development. The front-end view can use JavaScript or AngularJS to make a single page application that uses AJAX to call server-side components. Passwords shall be encrypted in Java and securely stored in the database. Users can upload a document or 4mage of their receipt when submitting reimbursements.

private Logger log = Logger.getRootLogger();

private UserService userService = new UserService();

protected void service(HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

super.service(request, response);

response.addHeader("Access-Control-Allow-Origin", "http://localhost:4200");

response.addHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS, PUT, DELETE, HEAD");

response.addHeader("Access-Control-Allow-Headers","Origin, Methods, Credentials, X-Requested-With, Content-Type, Accept");

response.addHeader("Access-Control-Allow-Credentials", "true");

response.setContentType("application/json");

}

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

if(request.getSession().getAttribute("user") == null) {

response.setStatus(403);

return;

}

List<User> users = new ArrayList<>();

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

users.add(new User(1, "blake","pass"));

ObjectMapper om = new ObjectMapper();

String json = om.writeValueAsString(users);

response.getWriter().write(json);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

// using JSON

String json = request.getReader().lines().reduce((acc, cur) -> acc + cur).get();

log.trace("json " + json);

ObjectMapper om = new ObjectMapper();

User credentials = (User) om.readValue(json, User.class);

log.trace(credentials);

User u = userService.login(credentials.getUsername(), credentials.getPassword());

// for default form submit method

// log.trace("post request made to login servlet");

// log.trace("username = " + request.getParameter("username"));

// log.trace("password = " + request.getParameter("password"));

// User u = userService.login(request.getParameter("username"),

// request.getParameter("password"));

if (u != null) {

HttpSession sess = request.getSession();

sess.setAttribute("user", u);

String respjson = om.writeValueAsString(u);

response.getWriter().write(respjson);

} else {

response.setStatus(401);

}

}

}