PeoplePlus
Project Plan

Version 1.2

Revision History

Date	Version	Description	Author
9/13/2019	1.0	Initial Plan, Milestones and S/W stack	Manmeet Singh
12/1/2019	1.1	Updated Development process, testcases	Jeyasri Subramanian
12/14/2019	1.2	Review and updated development process and execution	Subarna Chowdhury Soma

Table of Contents

1. Introduction

1.1 Purpose of this document

The purpose of this document is to provide a detailed project description of the application called *PeoplePlus*, which is designed to be a platform for any organization to manage their Human Capital. More specifically, it is designed to help companies on/off board employees, manage their training compliance. The document covers the design, test and deployment plan of the project. This document guides devOps for further maintenance and improvement of the project.

1.2 Intended Audience

This document shall be used in all phases of the project as a guideline. Intended audiences of this project are all project stakeholders:

- project supervisor
- project leader
- team members
- tester

1.3 Scope

This document defines the project plan of the *PeoplePlus* application. The overview includes objectives of the project, organization of the project team, development process that is going to be used during the project, assessment of possible risks, communication used between project stakeholders and project plan that includes time schedule and activity plan.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
PeoplePlus	The name of the project
Project Supervisor	A person in charge of supervising the project
Project Leader	A person in charge of organizing the team and communicating with the project supervisor
Team Member	An active member of the team responsible for making the job done
Tester	An active member of the team responsible for testing the application
Milestone	A time in a project that marks the end of a project phase or the completion of an important deliverable.
Git	Version control system that will be used in this project
Scrum	An iterative and incremental agile software development method for managing software projects and product or application development
Scrum sprint	The basic unit of development in Scrum
Scrum master	Ensures the smooth working of the Scrum team and enforces Scrum practices

Product owner	Responsible for product management and its quality

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
HRM	Human Resources Management
SSO	Single sign on

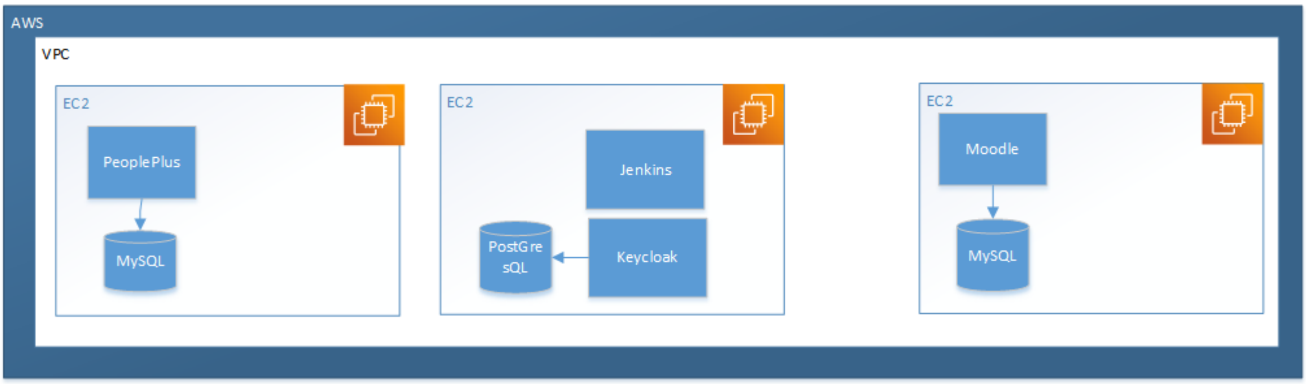
2. Objectives and Requirements

A company or organization's HR department is usually responsible for creating, putting into effect and overseeing policies governing workers and the relationship of the organization with its employees. HRM is a wide area and key department in an organization that governs the quality and delivery of an organization. The objective of this project to facilitate HR to govern resources between departments, promote, terminate, recruit and assign training to employees as per Manager's request. The key feature of this application is light-weight and integrated with third-party open source Moodle, which allows HR to create new course as per organization's need, and assign training to employees periodically. HR can move employees within their organization to different departments. HR can assign different learning specific role also for different training. The roles and titles of the employees in all departments are maintained in the same way.

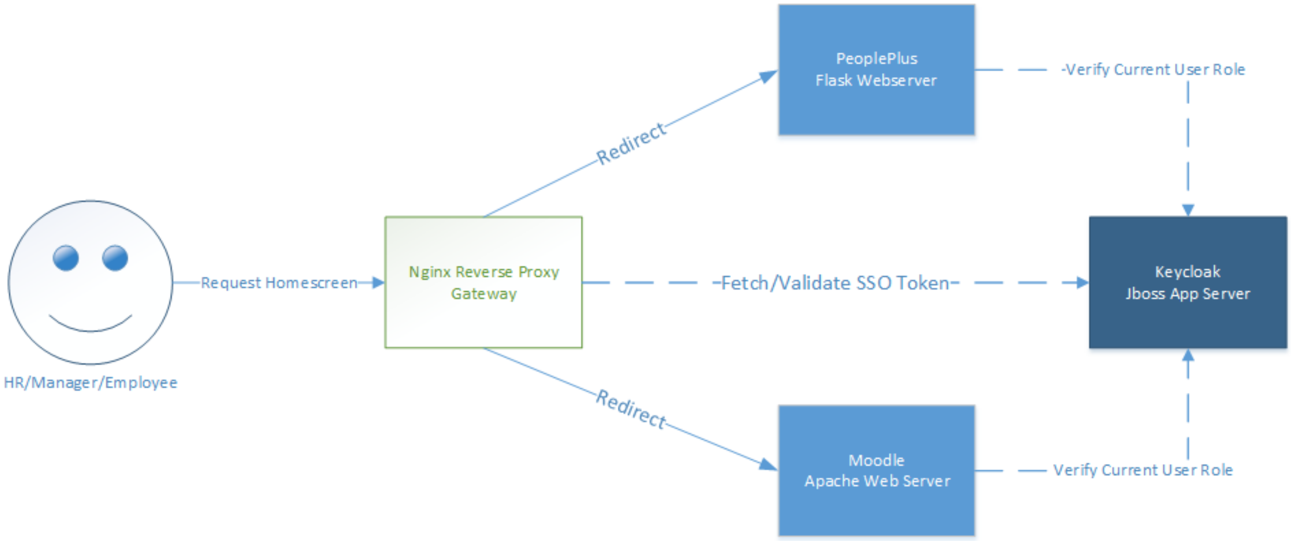
Our requirements for this project are to demonstrate:
- Single Sign On via HTTPS
- Employee Management
- Role Based Authentication
- Role Based Functionality
- Open source integration via Moodle
- Build a distributed system which communicates via REST APIs

3. Architecture & High Level Design

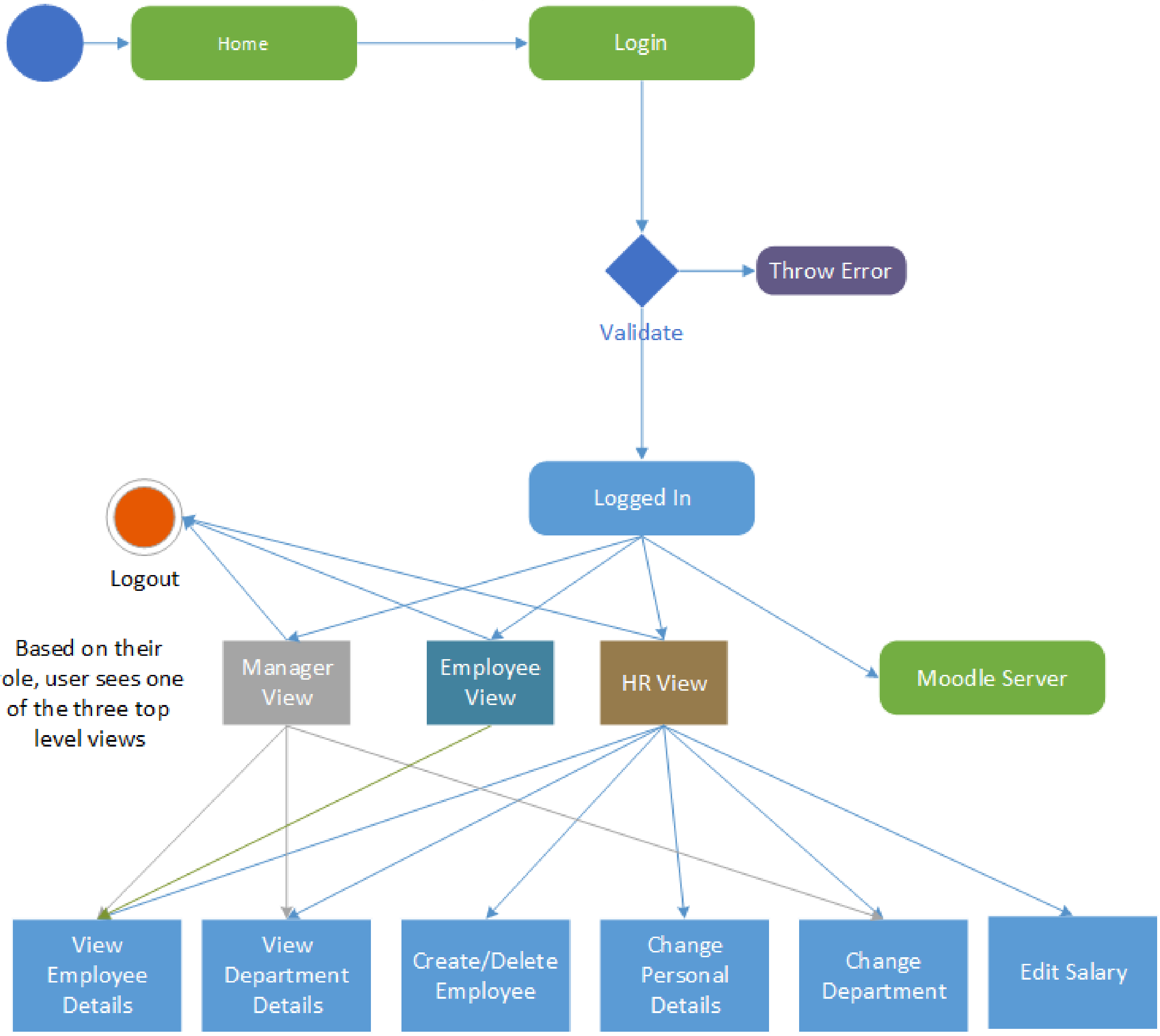
I. Container Level Design



II. Role Based SSO Design

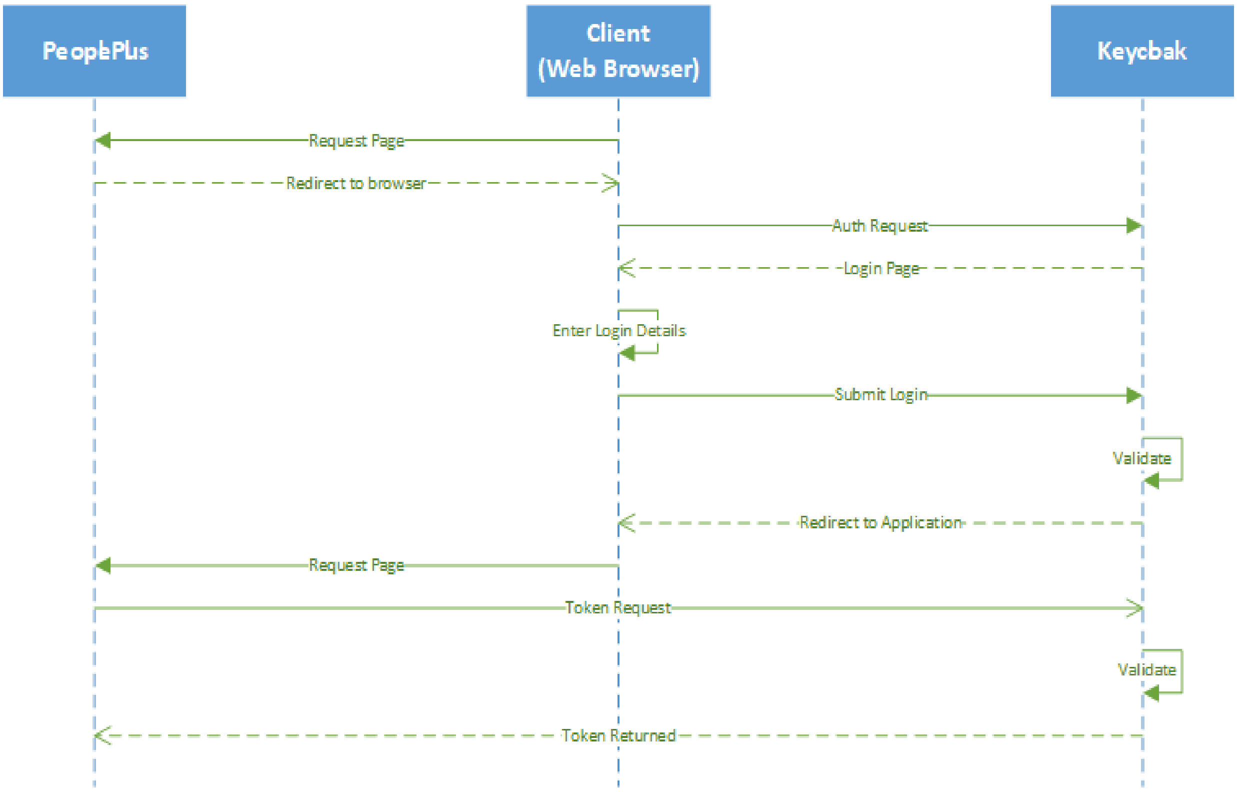


III. PeoplePlus (Homepage) Activity Diagram

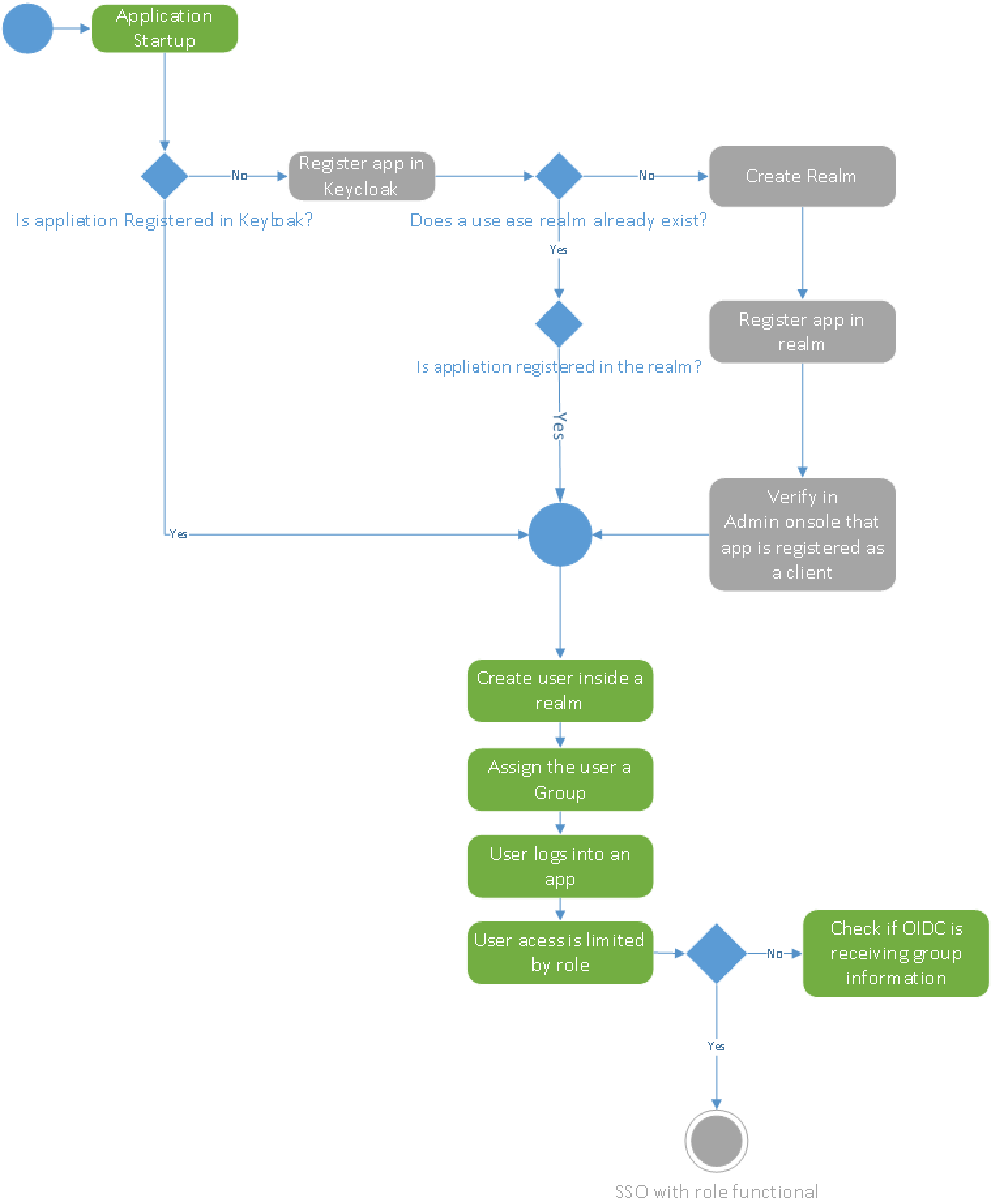


Based on their role, user sees one of the three top level views

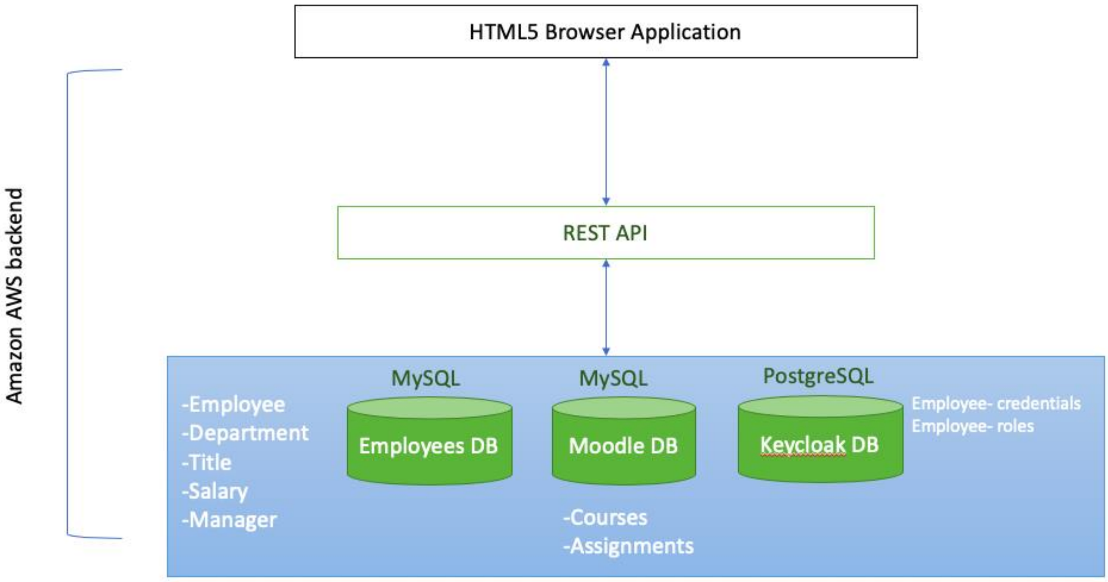
IV. Keycloak Login Sequence Diagram



V. Keycloak Configuration

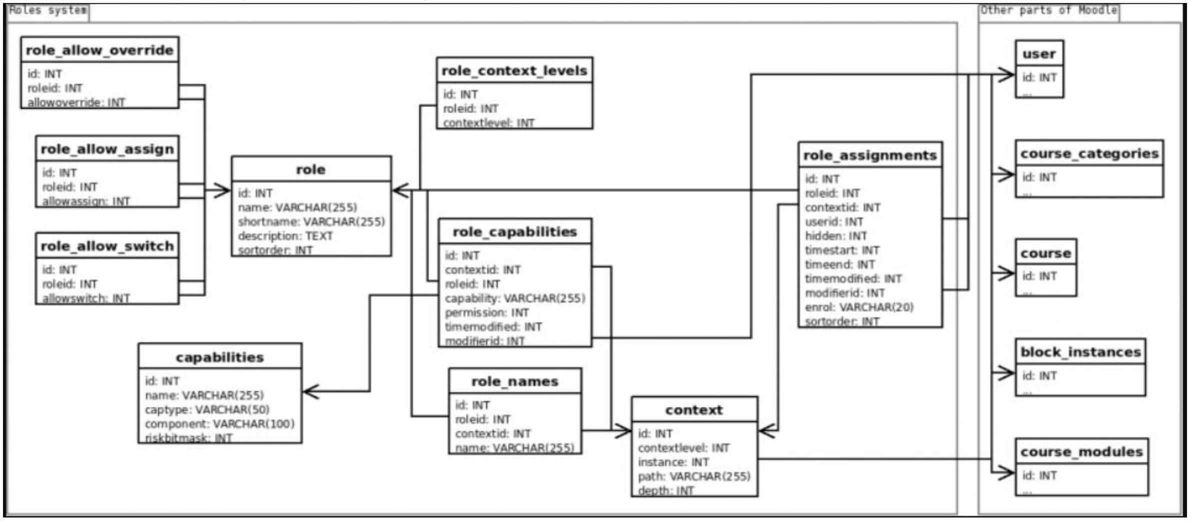

Application Startup
Is appliation Registered in Keycloak?
No
Register app in Keycloak
Does a use ase realm already exist?
No
Create Realm
Yes
Is appliation registered in the realm?
Register app in realm
Yes
Verify in Admin onsole that app is registered as a client
Yes
Create user inside a realm
Assign the user a Group
User logs into an app
User acess is limited by role
No
Check if OIDC is receiving group information
Yes
SSO with role functional

VI. Database Design



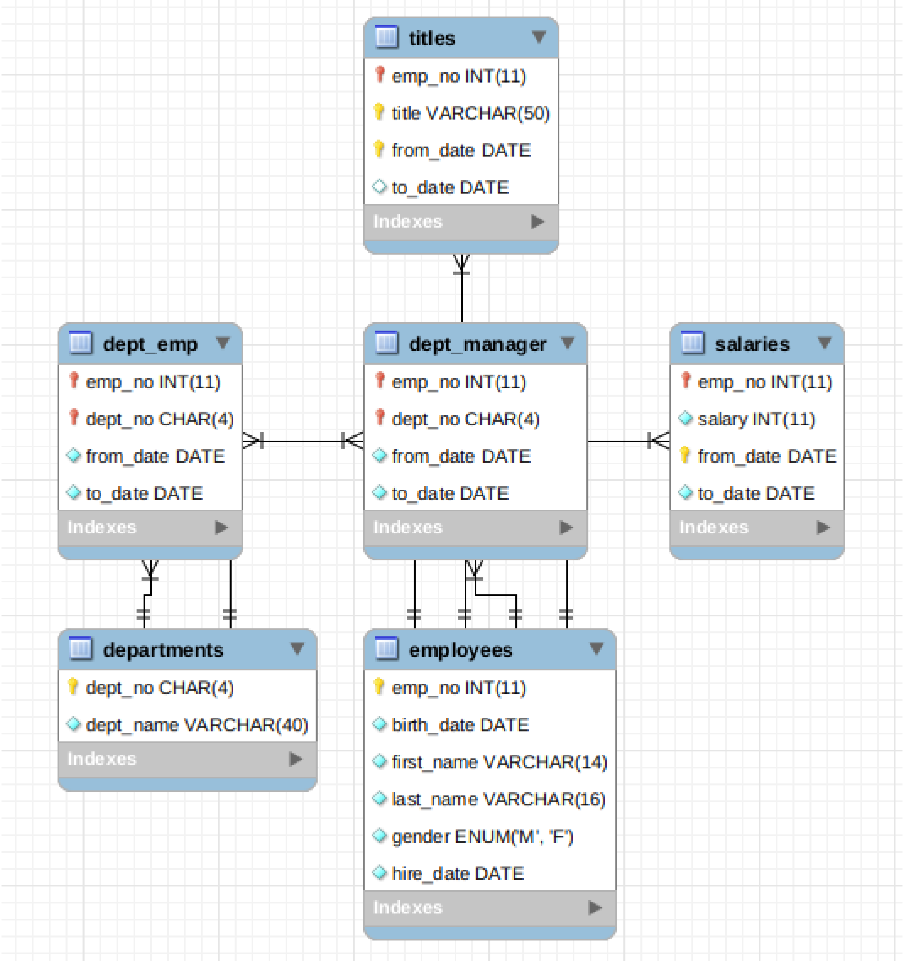
*REST API box represents the application layer

VI. Moodle role synch with our system:

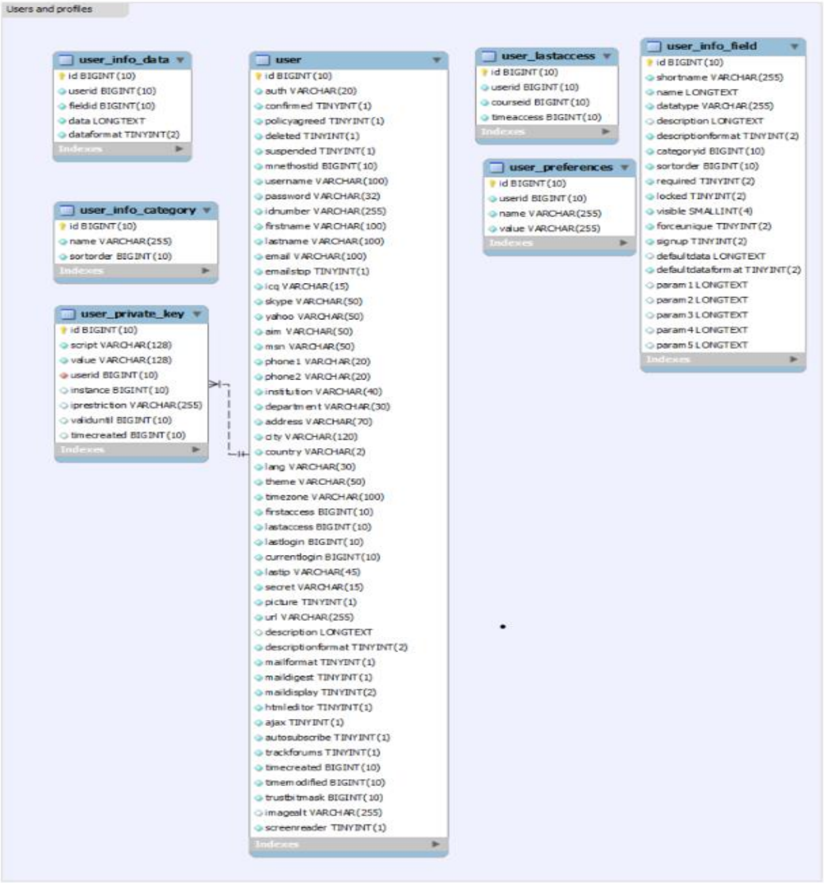


VII. SQL Relational Design

a. PeoplePlus Employee database



b. Moodle User profile:



user_info_data
id BIGINT(10)
userid BIGINT(10)
fieldid BIGINT(10)
data LONGTEXT
dataformat TINYINT(2)
Indexes

user_info_category
id BIGINT(10)
name VARCHAR(255)
sortorder BIGINT(10)
Indexes

user_private_key
id BIGINT(10)
script VARCHAR(128)
value VARCHAR(128)
userid BIGINT(10)
instance BIGINT(10)
iprestriction VARCHAR(255)
validuntil BIGINT(10)
timecreated BIGINT(10)
Indexes

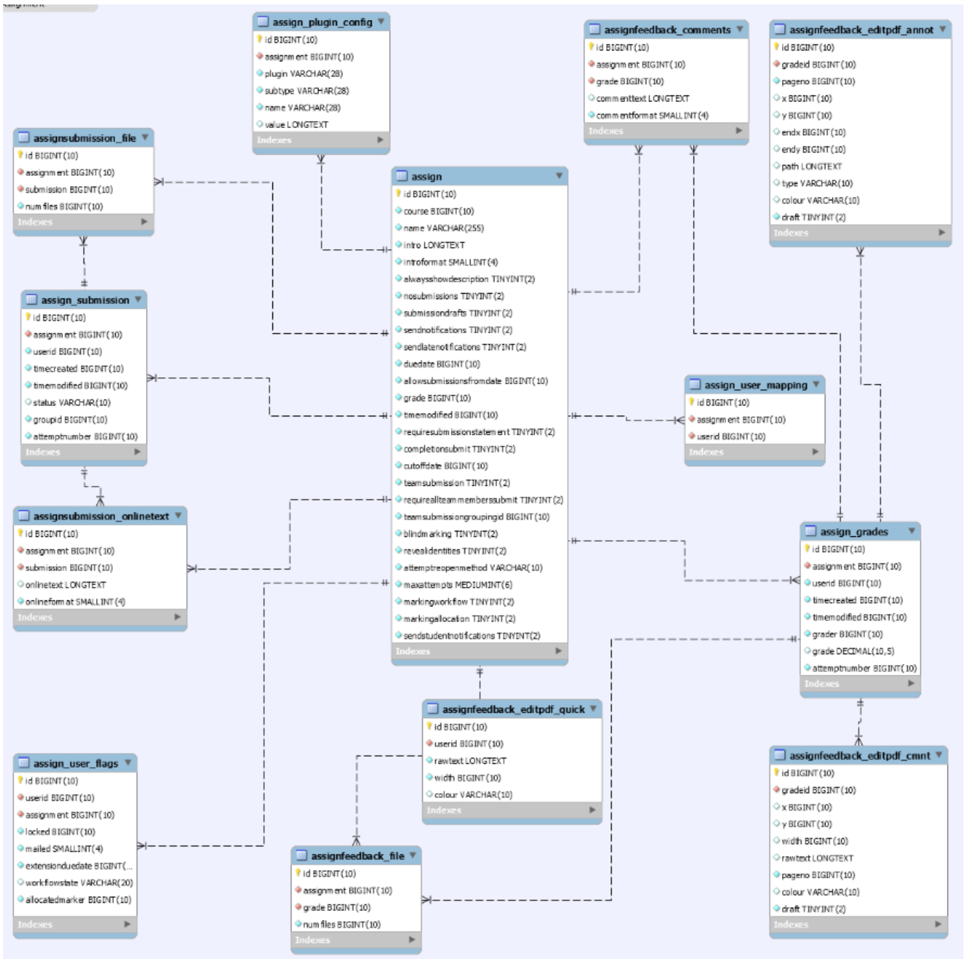
user
id BIGINT(10)
auth VARCHAR(20)
confirmed TINYINT(1)
policyagreed TINYINT(1)
deleted TINYINT(1)
suspended TINYINT(1)
mnethostid BIGINT(10)
username VARCHAR(100)
password VARCHAR(32)
idnumber VARCHAR(255)
firstname VARCHAR(100)
lastname VARCHAR(100)
email VARCHAR(100)
emailstop TINYINT(1)
icq VARCHAR(15)
skype VARCHAR(50)
yahoo VARCHAR(50)
aim VARCHAR(50)
msn VARCHAR(50)
phone1 VARCHAR(20)
phone2 VARCHAR(20)
institution VARCHAR(40)
department VARCHAR(30)
address VARCHAR(70)
city VARCHAR(120)
country VARCHAR(2)
lang VARCHAR(30)
theme VARCHAR(50)
timezone VARCHAR(100)
firstaccess BIGINT(10)
lastaccess BIGINT(10)
lastlogin BIGINT(10)
currentlogin BIGINT(10)
lastip VARCHAR(45)
secret VARCHAR(15)
picture TINYINT(1)
url VARCHAR(255)
description LONGTEXT
descriptionformat TINYINT(2)
mailformat TINYINT(1)
maildigest TINYINT(1)
maildisplay TINYINT(2)
htmleditor TINYINT(1)
ajax TINYINT(1)
autosubscribe TINYINT(1)
trackforums TINYINT(1)
timecreated BIGINT(10)
timemodified BIGINT(10)
trustbitmask BIGINT(10)
imagealt VARCHAR(255)
screenreader TINYINT(1)
Indexes

user_lastaccess
id BIGINT(10)
userid BIGINT(10)
courseid BIGINT(10)
timeaccess BIGINT(10)
Indexes

user_preferences
id BIGINT(10)
userid BIGINT(10)
name VARCHAR(255)
value VARCHAR(255)
Indexes

user_info_field
id BIGINT(10)
shortname VARCHAR(255)
name LONGTEXT
datatype VARCHAR(255)
description LONGTEXT
descriptionformat TINYINT(2)
categoryid BIGINT(10)
sortorder BIGINT(10)
required TINYINT(2)
locked TINYINT(2)
visible SMALLINT(4)
forceunique TINYINT(2)
signup TINYINT(2)
defaultdata LONGTEXT
defaultdataformat TINYINT(2)
param1 LONGTEXT
param2 LONGTEXT
param3 LONGTEXT
param4 LONGTEXT
param5 LONGTEXT
Indexes

c. Moodle role and course assignment combination



4. Organization

4.1 Project group

Name	Initials	Responsibility (roles)
Snehal Yeole (TA)	SY	Project Supervisor
Manmeet Singh	MS	Project Leader
Jeyasri Subramanian	JS	Team Member
Subarna Chowdhury Soma	SC	Team Member

4.2 Customer

Our customers include organizations of any size looking to achieve operations efficiency in managing their Human Resources.

The broad set of target customers can be summarized as:

- IT companies
- Small-scale business units
- Manufacturing units

5. Development process

The project is developed in Python. APIs are managed by Flask Framework, role-based access id maintained using Keycloak open source tool. Learning modules are integrated and managed with Moodle third party framework. Moodle is connected with keycloak through open id plugin. Moodle API's are accessed by our python-based client service. Moreover, Moodle authentication and role assignment components code are modified to map our HR/Manager/Employee role with their system specific role and every employee by default gets a 'Student' role inside Moodle learning system. Employees and other related data for employee management and Moodle learning data is stored in MySQL DB. Keycloak single sign on and role management is stored in PostgresDB.

The overall development process we followed was agile alongside test driven development. We developed various components of the system starting with Single Sign-On. From that point onward, we kept track of tasks that everyone was working on, and the features giving them trouble through Google Docs. Additionally, we met on weekly basis to integrate and ensure that we didn't have any major compatibility issues.

6. Deliverables

To	Output	Planned week	Promised week	Late +/-	Delivered week	Notes
Customer	Standalone web app integrated with single sign on	09/6/19	9/27/19	0	9/27/19	
Customer	API to fetch department details and Employee details	9/6/19	9/27/19	1	9/29/19	
Customer	Single sign on with Role management	9/27/19	10/4/19	3	10/7/19	
Customer	Create and edit employee API	9/6/19	10/18/19	0	10/18/19	
Customer	Moodle Course creation API	9/6/19	9/27/19	1	9/29/19	
Customer	Moodle role management for HR and Employee roles	9/27/19	10/4/19	1	10/5/19	
Customer	Terminate Employee safe deletion	9/6/19	10/11/19	1	10/12/29	

Customer	Keycloak role management integration with employee management module	11/1/19	11/15/19	2	11/17/19	
Customer	Keycloak role management integrated with moodle module	11/1/19	11/22/19	7	11/29/19	
Customer	Writing unit Test cases	11/1/19	11/15/19	0	11/15/19	
Customer	Application testing	11/29/19	12/1/19	0	12/1/19	

7. Project risks

Possibility	Risk	Preventive action
Keycloak opensource integration with Flask app	High	Team members will communicate periodically and following tutorials about keycloak API
Keycloak opensource integration in moodle	High	Team members will communicate periodically and posted queries in Stackoverflow to understand prior developer experience
Moodle open source is developed with PHP, team members lack PHP programming skills	High	Team members will spend time on learning basic PHP syntax and we have identified a few python wrappers for this integration
Periodic code check and integration is needed	Medium	Team involved in continuous collaboration and each developer maintained separate branches in github to avoid conflict
Cloud deployment	Medium	AWS EC2 instance is not free and we continued to use for quick integration

8. Accomplishments

In this project, we accomplished creating a base HR Management platform which can be extended with many other modules that build on top of the basic infrastructure that we laid down. A few top level accomplishments are:

- Keycloak authentication via HTTPS
    - Setting up of self-signed certificates as trusted on clients using various certificate management back ends (java, python, bash, etc.)
- Integrating open source learning module- Moodle with our system
- Modifying Moodle source code to fully authenticate via Keycloak and to synch moodle's system role with our application's role [HR/Manager/Employee]
    - User roles are fetched from Keycloak as well, with all changes reflected.
- Implementing CRUD based on a 300k+ record database.
    - Managing CRUD permissions via role hierarchy

9. Communication

9.1 Collaboration

Team members are in continuous collaboration in Slack for any project development queries.
Meetings are conducted periodically in libraries and the virtual meetings via Google Hangouts.

9.2 Git

All source code with proper comments will be available in Github
Repository URL: https://github.com/manmeet3/cmpe272_PeoplePlus

The repository has multiple branches, each representing an application for us. We chose to use different branches for different projects in order to have one location for testing and integration. The various branches are:

- Master: This branch contains the main employee management app named PeoplePlus.
- Baseapp: This is a landing page app which performs functionality similar to https://one.sjsu.edu/. It is fully functional, but due to the time constraints, it was not recorded as part of the video.
- Keycloak: This branch holds the keycloak webserver and the BlueHats organization configuration. It's ready to run once java 8 is installed on a machine.
- Moodle_aws: This is our fork of the Moodle open learning management application. We worked on this branch to modify the source code to make it compatible with our workflow.

10. Project plan

10.1 Time schedule

Id	Milestone Description	Responsible Dept./Initials	Finished week				Metr.	Rem.
			Plan	Forecast		Actual		
				Week	+/-			

	Identify SSO client	MS	9/8	9/8	-1	9/6		
	Identify third party tools that facilitate training to employees	SC	9/8	9/8	+2	9/10		
	Setup backend database	MS	9/15	9/15	0	9/15		
	Setup github repository	MS	9/15	9/15	0	9/15		
	Setup development environment by each team member	JS	9/8	9/8	+3	9/11		
	Create login page .html. css files	JS	9/13	9/13	-1	9/12		
	Integrate SSO for one user with username: admin, password: test	MS	9/29	9/29	+1	9/30		
	Setup initial flask framework with one GET API working	JS	9/29	9/29	+2	10/01		
	Download and verify moodle in localhost	SC	9/15	9/15	0	9/15		
	Create and verify roles and groups in keycloak	MS	10/13	10/13	-1	10/12		
	Implement create and edit employee API	JS	10/6	10/6	+2	10/8		
	Implement terminate API	JS	10/13	10/13	0	10/13		
	Integrate keycloak role mapping with API	MS	10/6	10/6	+2	10/8		
	Implement API	SC	10/6	10/6	+3	10/9		

	to create a user in moodle							
	Implement moodle api to create course	SC	10/13	10/13	-1	10/12		
	Implement moodle api for role authorization	SC	10/20	10/20	+1	10/20		
	Integrate moodle role with Keycloak role management	MS/ SC	10/27	10/27	+3	10/30		
	Edit employee API to update Department	JS	10/27	10/27	+3	10/30		
	Integrate Keycloak with employee management app	JS	11/10	11/10	+1	11/11		
	Update all employee API with Role management	JS	11/17	11/17	+1	11/18		
	Create test cases for API	JS	11/24	11/24	+2	11/26		
	Implement Keycloak API to create user account and role for new employee	JS	11/17	11/17	0	11/17		
	Deploy testcases with Jenkins	MS	12/01	12/01	+2	12/03		

10.2 Test plan

Test No.	001	**Phase:**	1	**Author:**	JS	Date: 11/29
Test Category:		**Unit Testcase**				
Software Product:		Flask-unittest				

Test Title:	Test Home Page
Test Purpose:	Verify whether User can access the application
Test Setup:	CreateTestApp in setUp()
Prerequisites:	Source code should be present in the same folder path
Procedure:	Execute test cases from command line
Checks:	**The unit case creates a app instance and verifies home page**
Expected Results:	PASS
Result:	PASS
Reason for Failure:	
Remarks:	

Test No.	002	**Phase:**	1	**Author:**	JS	Date: 11/29
Test Category:		**Unit testcase**				
Software Product:		Flask -unittest				
Test Title:		Verify view departments				
Test Purpose:		Verify whether HR able to view departments				
Test Setup:		Create TestApp and connect to test db in setUp()				
Prerequisites:		Source code should be present in the same folder path				
Procedure:		Execute testcase from command line				
Checks:		**The testcase verifies list department api**				
Expected Results:		PASS				
Result:		PASS				
Reason for Failure:						
Remarks:						

Test No.	003	**Phase:**	1	**Author:**	JS	Date: 11/29
Test Category:		**Unit testcase**				
Software Product:		Flask- unittest				
Test Title:		Verify 404 status code				
Test Purpose:		Verify whether the application communicates failure code to user				
Test Setup:		Create TestApp and connect to test db in setUp()				
Prerequisites:		Source code should be present in the same folder path				
Procedure:		Testcase verifies whether all incorrect navigation to correct home page				
Checks:						
Expected Results:		PASS				
Result:		PASS				
Reason for Failure:						
Remarks:						

Test No.	004	**Phase:**	1	**Author:**	JS	Date: 11/29
Test Category:		**Unit testcase**				
Software Product:		Flask unittest				
Test Title:		Verify 403 status code				
Test Purpose:		Verify whether users with inappropriate role renders to 403 status code				
Test Setup:		Create TestApp and connect to test db in setUp()				
Prerequisites:		Source code should be present in the same folder path				
Procedure:		Testcase verifies whether user has correct permission to access the page				
Checks:						
Expected Results:		PASS				
Result:		PASS				

Reason for Failure:	
Remarks:	

Test No.	005	**Phase:**	1	**Author:**	JS	Date:
Test Category:		**Unit testcase**				
Software Product:		Flask unittest				
Test Title:		Verify 500 status code				
Test Purpose:		Check whether API fails to connect to DB is handled correctly				
Test Setup:		Create testapp with setUp()				
Prerequisites:		Cloud db is not running				
Procedure:		Testcase verifies whether lost DB connection is handled correctly				
Checks:						
Expected Results:		PASS				
Result:		PASS				
Reason for Failure:						
Remarks:						

11. References

1. https://scotch.io/courses
2. https://aws.amazon.com/getting-started/tutorials/create-microsoft-sql-db/
3. https://www.palletsprojects.com/p/flask/
4. https://developers.redhat.com/blog/2018/03/19/sso-made-easy-keycloak-rhsso/
5. https://www.keycloak.org/downloads.html
6. https://github.com/mitraining/moodle-open-id-connect-authentication-plugin
7. https://github.com/moodle/moodle
8. https://docs.moodle.org/dev/Web_service_API_functions
9. https://docs.moodle.org/38/en/Using_web_services
10. https://docs.moodle.org/dev/Database_schema_introduction
11. https://docs.moodle.org/20/en/index.phptitle=Image:Users_and_profiles_erd.png&amp%3Bdiff=0&amp%3Boldid=prev