INTRODUCTION

The major idea behind this application is to allow enterprises such as schools, colleges, vital day-to-day processes such as employee management, faculty management, meeting and schedule management, analytics and reporting. Most colleges keep track of their tasks, manually (books, files). This is a manual process and often takes a lot of time, it is difficult to maintain and is clearly a wastage of paper. Social media or other time management tools provide limited functionality and are not customizable. While there are tools that provides options to log details or activities, they are often difficult to use and are not user friendly. The application consists of two main components. One is the web application and the other being the mobile application. The web application is responsible for authentication, user role management and database management whereas the mobile application is responsible for users' attendance or presence via scan-to-confirm strategy. It provides customizable options for creating users with role management and secure authentication. The schedule of activities are recorded by the administrator at the beginning of the session. The mobile app is responsible for recording the actual events occurred when the session is held. It records the time, date and location of the incident by a QR Code and sends the data securely to the web application which in turn stores and processes the information. Reports can be generated based on processed information and made available to different users based on their roles. Whenever there is a change in schedule, it will be notified to the faculty through the app.

The project is built keeping scalability in mind. The selection of tools and technologies for the project helps us in serving millions of users, including real-time data without any delays. The application is containerized into microservices where a load balancer is attached that handles a heavy amount of traffic. Also, continuous integration and continuous delivery techniques of deploying and testing the project has been implemented for better developer operations. The project is currently under the development phase where it is being tested for accuracy and reliability. It will help every enterprise in making their day to day processes easier and accountable. Further plan is to deploy the product in any enterprise organization which strive for quality.

ANALYSIS

Literature survey:

There are many number of applications which are both free and proprietary. The process of building this application had references to the following applications.

Replicon - is a configurable time and attendance platform enables your organization to manage dispersed workforce. Employees can provide inputs from the field using our advanced mobile app. Supervisors gain complete control over employee availability, time off and scheduling.

Jibble - Manage time & attendance for team. Employees can clock in, punch in or as we say, jibble in and out using the iPad Kiosk, Web or Mobile (iOS & Android). Work hours are accurately captured including activities and notes. Generate automated timesheets, activity/project tracking, client billing and powerful reporting for your team. Whether you are running a tech startup, consulting firm, or running the local restaurant, Jibble helps you with payroll, billing or team productivity

TrackerPal - Time and Attendance Software with Geofencing. Easily manage remote and field employee attendance. Employees punch in and out using their mobile phone. Optional selfie. You can specify a location for attendance. Supports leave request and approval.

TimeCamp - In its mobile version, TimeCamp helps users to track time automatically to specific projects, whether new or existing ones, and change the time entries manually. This attendance tracking software can also be set up on Android and iPhone and other mobile devices which run the iOS platform.

Many of these applications are built focusing employee management and not solely around attendance management. This applications focus is to make the attendance management and generate a report which can be used to improve the efficiency of the organization.

Functional requirements:

- **Admin:** Admin should be able to create departments and assign respective chairperson.
- **Report:** Chairpersons should be able to generate and analyze the weekly, monthly and semester wise report
- **Authorization:** Only the authorized admin should be able to login to the application where he/she can use the application.
- **Role management:** Admin can have one more role as a user/faculty. He/she should be assigned proper role.
- **Schedule management:** Admin should be able to make changes in the existing time table if there is a need.

Nonfunctional requirements:

- **Scalability:** Application is containerized for traffic distribution and scalability.
- **Maintainability:** Distributed version control and task management Continuous integration and continuous delivery for seamless deployment
- **Screen compatibility:** Works across devices or any screen via a browser
- **Platform independence:** The application should run on both the popular operating systems Windows, Ubuntu etc.
- **Performance:** The application should be able to run on all versions of operating systems after a limit and should consume less power. The application should not crash at any condition.
- **Permissions and Authentication:** The application should ask for user's permissions to access internet, send notifications etc. Anyone apart from the authenticated user should not be able to log in. If someone tries to penetrate the application, the tokens generated are verified using libraries like JWT.
- **Data Integrity:** Data is store in multiple nodes as a backup in case of emergency or recovery

Software requirements:

- **Backend**:
  - Node.js v10.x, Express.js v8.x
- **Frontend**:
  - React.js v16.x, Redux v16.x
- **Database:**
  - MongoDB v4.x
- **Testing**:
  - Mocha, Postman
- **DevOps**:
  - Github, Travis CI, AWS, Heroku
- **Other Tools**:
  - Draw.io, Balsamiq

Tools and Technologies:

Application Development technologies:

- This application is built using MERN stack i.e. MongoDB, Express, React Native, NodeJS. Redux and Firebase are also used to build this application.
- MongoDB is an open-source database software which is NoSQL in architecture. It stores data as JSON documents. It is fast, reliable and partition tolerant.
- Express is a web application framework for Node.js. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js
- React native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android.
- Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser.
- Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to Facebook's Flux architecture
- Firebase is a mobile and web application back-end as a service development platform

All these technologies are open-source and has wide community support. MERN stack is very popular, growing is usage and developer friendly.

Integration tools:

Travis-CI:

Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub. Open source projects may be tested at no charge via travis-ci.org. Private projects may be tested at travis-ci.com on a fee basis.

Git and GitHub:

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

GitHub is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features.

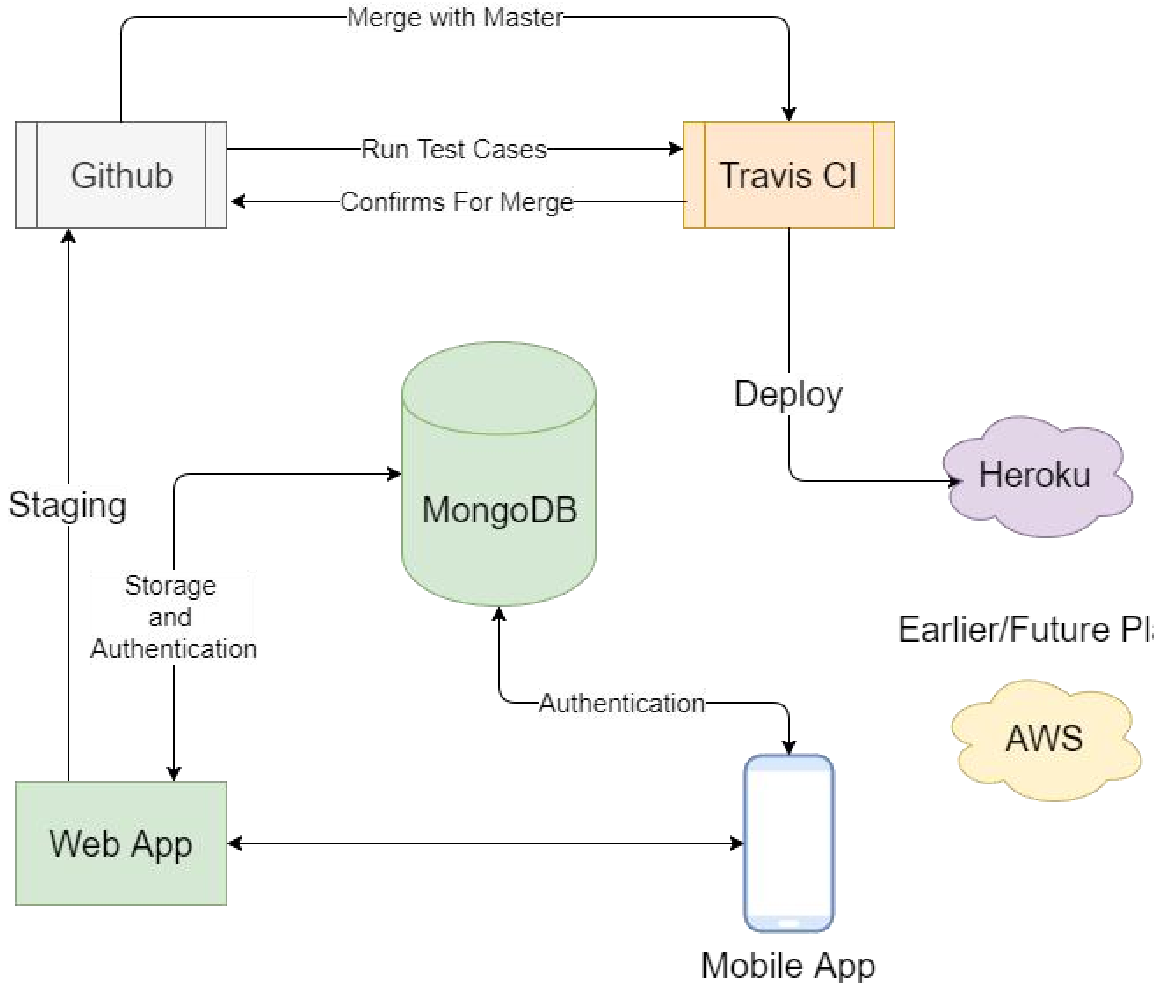
Since ‘Vibhaag mobile app’ is a companion of web app, it is developed on the ‘development branch’ and then integrated to master branch where it is tested.

Visual Studio Code:

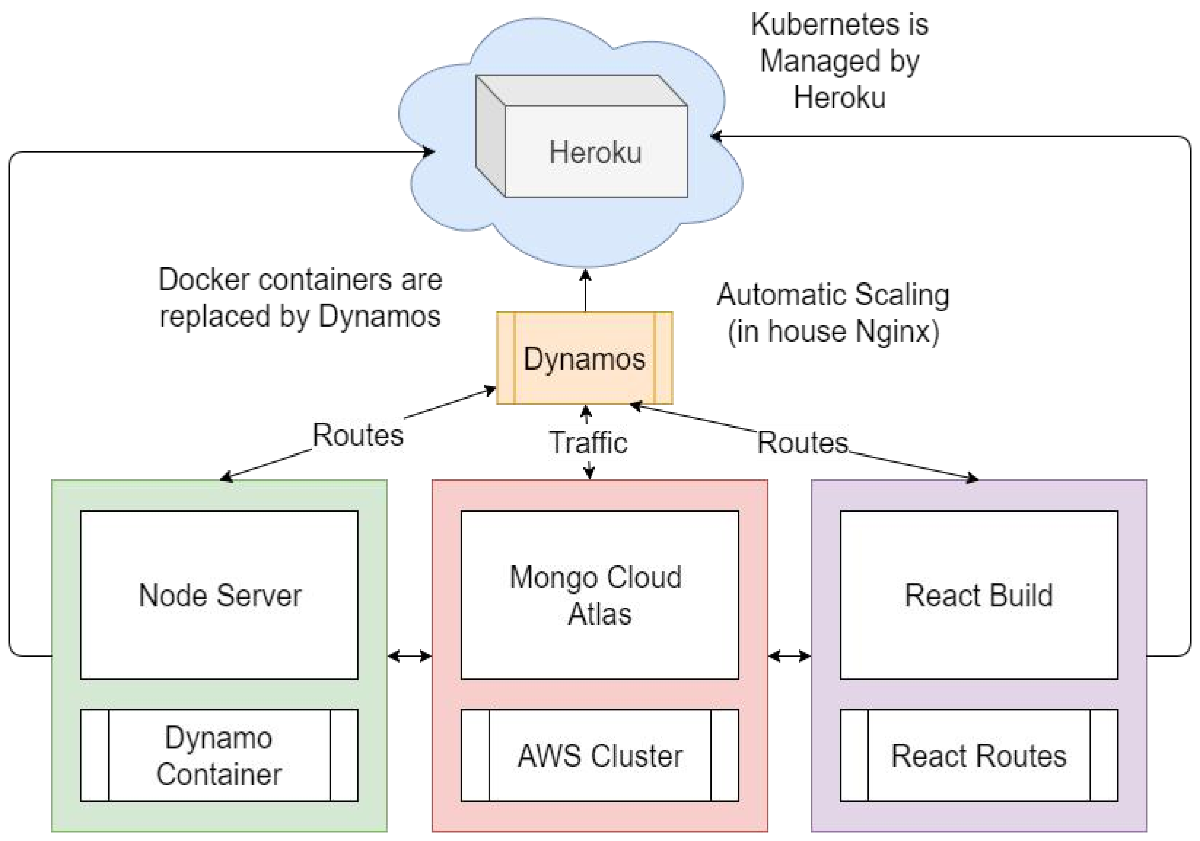
Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.
Source code is generated on VS Code and compiled in Android Studio. It is also tested on the android and device while developing

DESIGN

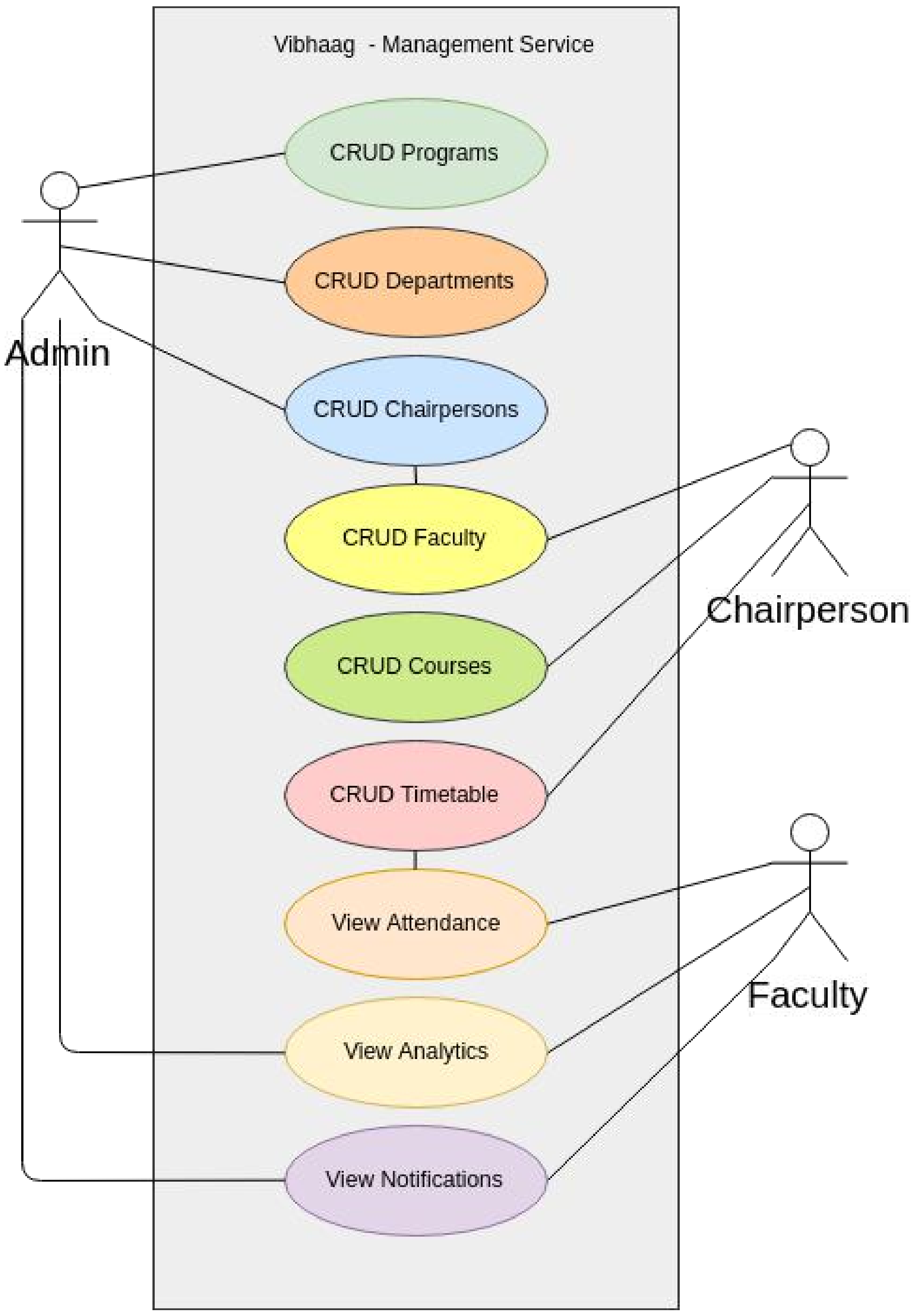
Process flow:



System Design:

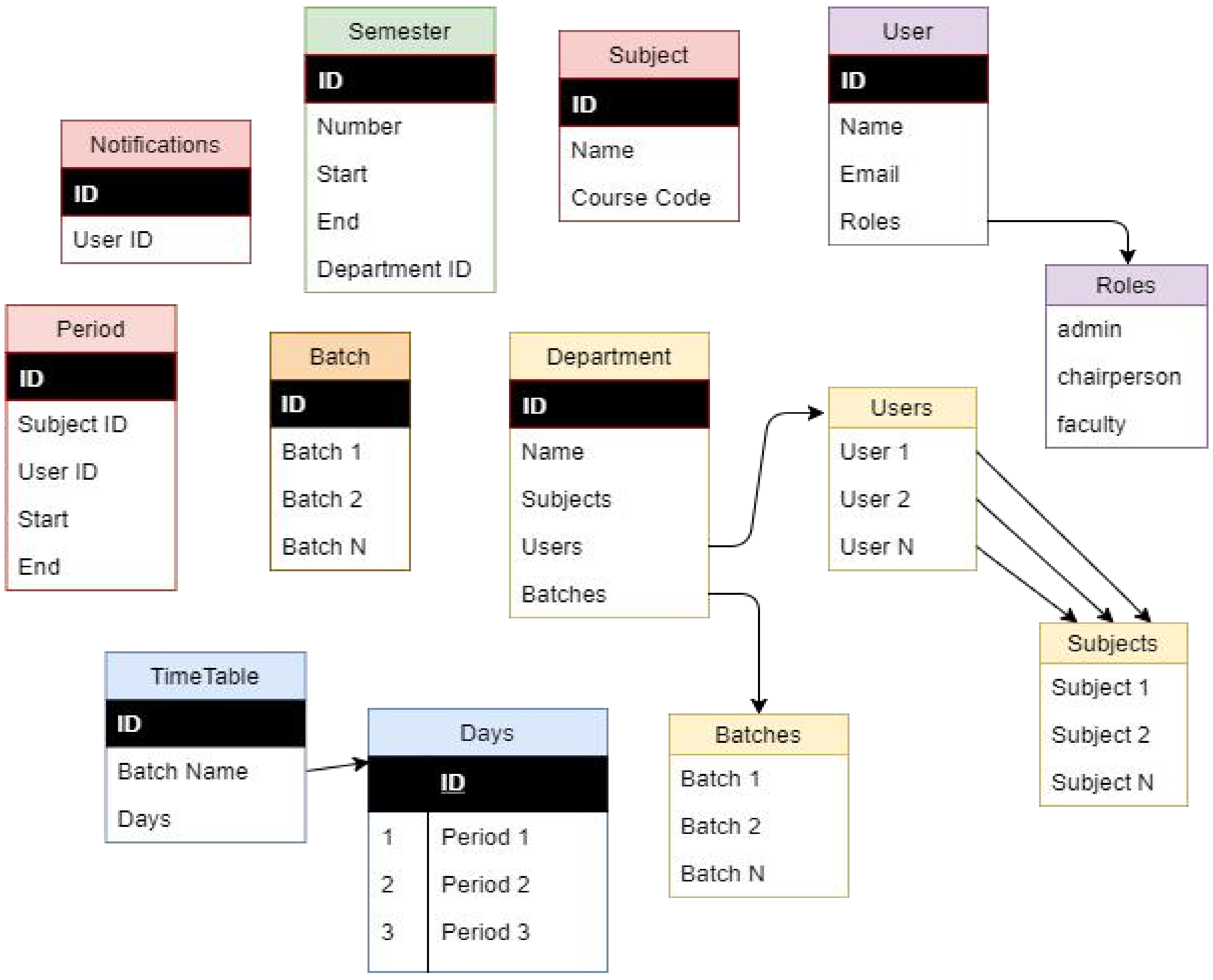


Use cases:



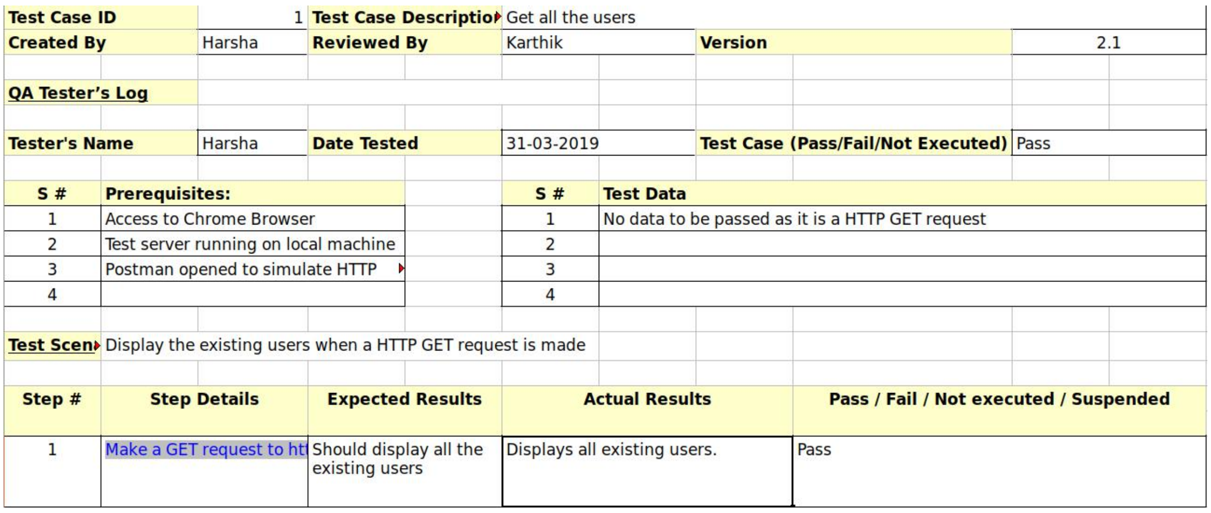
Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform

Database structure:



Our database encompasses a wide variety of different technologies that were developed in response to the demands presented in building modern applications.

TESTING

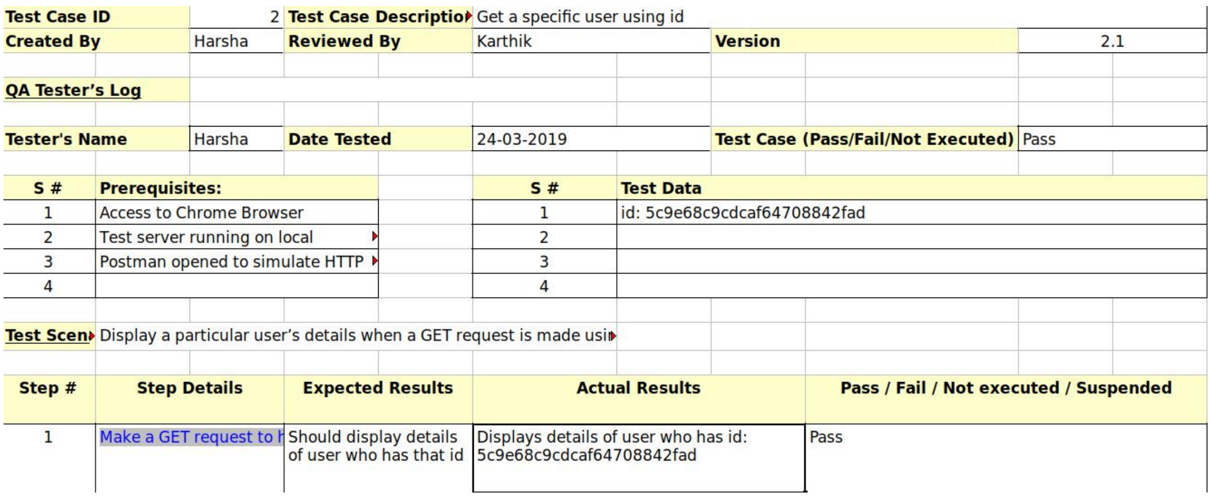


Test Case ID	1	**Test Case Descriptio**	Get all the users		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1
QA Tester's Log					
Tester's Name	Harsha	**Date Tested**	31-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	No data to be passed as it is a HTTP GET request
2	Test server running on local machine	2	
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Display the existing users when a HTTP GET request is made

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Make a GET request to htt	Should display all the existing users	Displays all existing users.	Pass

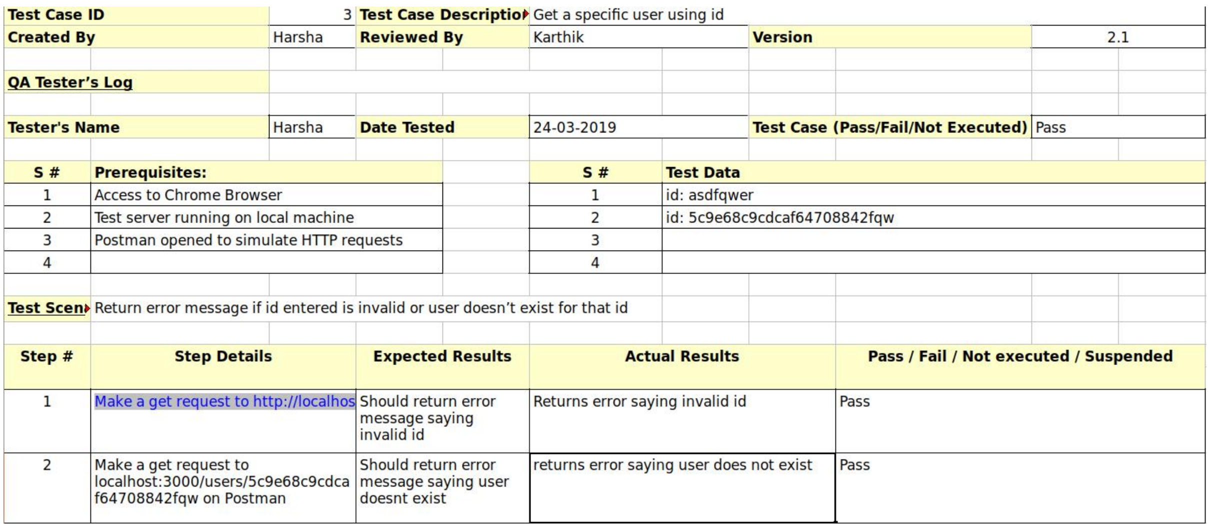


Test Case ID	2	**Test Case Descriptio**	Get a specific user using id		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1
QA Tester's Log					
Tester's Name	Harsha	**Date Tested**	24-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	id: 5c9e68c9cdcaf64708842fad
2	Test server running on local	2	
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Display a particular user's details when a GET request is made usi

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Make a GET request to h	Should display details of user who has that id	Displays details of user who has id: 5c9e68c9cdcaf64708842fad	Pass



Test Case ID	3	**Test Case Descriptio**	Get a specific user using id		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1
QA Tester's Log					
Tester's Name	Harsha	**Date Tested**	24-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	id: asdfqwer
2	Test server running on local machine	2	id: 5c9e68c9cdcaf64708842fqw
3	Postman opened to simulate HTTP requests	3	
4		4	

Test Scen Return error message if id entered is invalid or user doesn't exist for that id

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Make a get request to http://localhos	Should return error message saying invalid id	Returns error saying invalid id	Pass
2	Make a get request to localhost:3000/users/5c9e68c9cdcaf64708842fqw on Postman	Should return error message saying user doesnt exist	returns error saying user does not exist	Pass

Test Case ID	4	**Test Case Descriptio**	Delete a specific user using id		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1
QA Tester's Log					
Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	id: 5c9e68c9cdcaf64708842fad
2	Test server running on local	2	
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Delete a user's info when a DELETE request is made using a

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Make a DELETE request	Should delete the user's details after request is made	Deletes the user's details whose id is 5c9e68c9cdcaf64708842fad	Pass

Test Case ID	5	**Test Case Descriptio**	Delete a specific user using id		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1
QA Tester's Log					
Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	id: asdfqwer
2	Test server running on local	2	id: 5c9e68c9cdcaf64708842fqw
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Return error message when an invalid id is entered or user o

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Make a DELETE request	Should return error message saying invalid id	Returns error message saying invalid id	Pass
2	Make a DELETE request to localhost:3000/users/5c9e68c9cdcaf64708842fqw on Postman	Should return error message saying user doesn't exist	Returns error message saying user does not exist	Pass

Test Case ID	6	**Test Case Descriptio**	Create a new user		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	name: Harsha Ky
2	Test server running on local	2	email: harsha@gmail.com
3	Postman opened to simulate HTTP	3	password: userpassword
4		4	roles: admin

Test Scen Create a user after taking details from the HTML body

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter details of user in the body	no result expected	No result got	Pass
2	Make a POST request to	Should create a new user if a user with same email doesn't exist	Created new user	Pass

Test Case ID	7	**Test Case Descriptio**	Create a new user		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	name: Harsha Ky
2	Test server running on local	2	email: asdfqwer1234
3	Postman opened to simulate HTTP	3	password: userpassword
4		4	Roles: 1234

Test Scen do not create user when wrong / invalid info is sent

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter wrong details of user in the body	no result expected	No result got	Pass
2	make a POST request to localhost:3000/users	should return error message saying wrong/invalid information	Returns error message saying wrong/invalid information	Pass

Test Case ID	8	**Test Case Descriptio**	Login user if entered info is correct		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	email: harsha@gmail.com
2	Test server running on local	2	password: userpassword
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Login a user if the entered credentials are correct

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter details of user in the body	no result expected	No result got	Pass
2	Make a POST request to	Should create an x-auth token and return back the token as header and save the token and login the user	Created an x-auth token and returns back the header and saves the token and logs in the user.	Pass

Test Case ID	9	**Test Case Descriptio**	Login user if entered info is correct		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	email: harsha@gmail.com
2	Test server running on local	2	password: asfwrq21r4
3	Postman opened to simulate HTTP	3	
4		4	

Test Scen Login a user if the entered credentials are correct

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter details of user in the body	no result expected	No result got	Pass
2	Make POST request to localhost:3000/users/login	Should return error message saying entered details are incorrect and should not create an x-auth token	Returns error message saying entered details are wrong and did not create an x-auth token	Pass

Test Case ID	10	**Test Case Descriptio**	Update a user's info		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	email: harsha@gmail.com
2	Test server running on local	2	name: Harsha Ky
3	Postman opened to simulate HTTP	3	id: 5c9e68c9cdcaf64708842fad
4		4	

Test Scen Update a user's info and send back the updated user info

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter the things a user wants to update	no result expected	No result	Pass
2	Make a PUT request to h	Should update the info of the user to whom the id belongs and return back the updated info of the user.	Updates the info of the user with id 5c9e68c9cdcaf64708842fad and returns back the updated info of the user.	Pass

Test Case ID	11	**Test Case Descriptio**	Update a user's info		
Created By	Harsha	**Reviewed By**	Karthik	**Version**	2.1

QA Tester's Log

Tester's Name	Harsha	**Date Tested**	28-03-2019	**Test Case (Pass/Fail/Not Executed)**	Pass

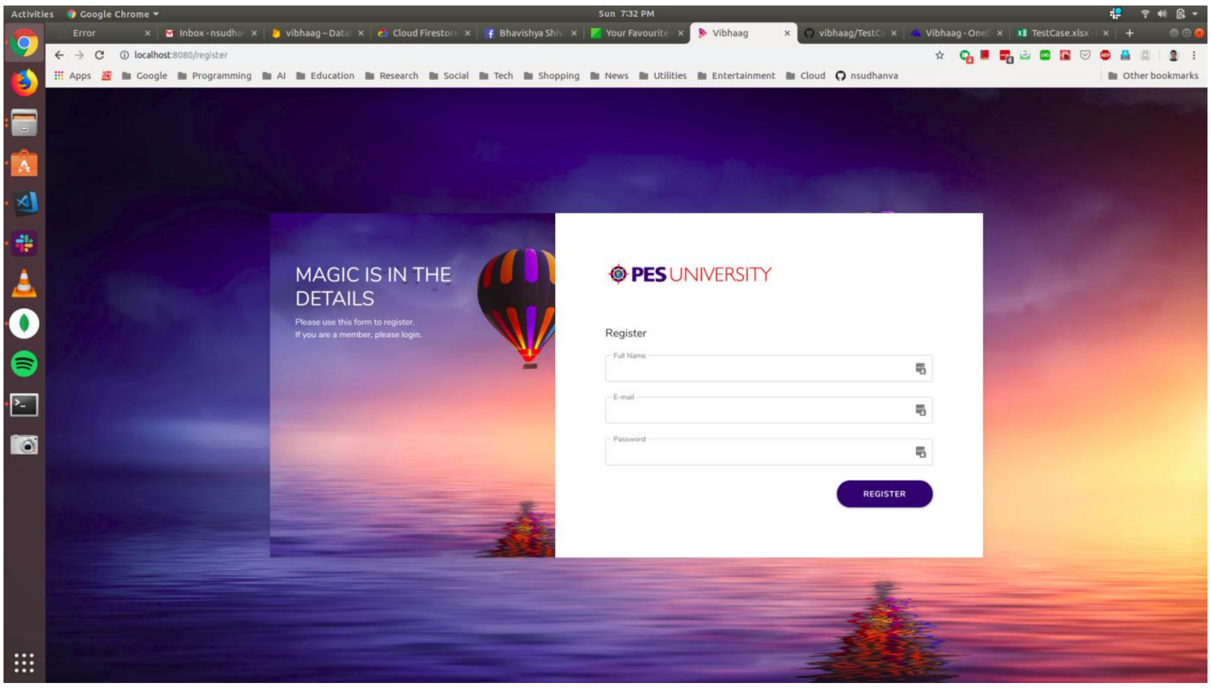
S #	**Prerequisites:**	**S #**	**Test Data**
1	Access to Chrome Browser	1	email: harsha@gmail.com
2	Test server running on local	2	name: asdf
3	Postman opened to simulate HTTP	3	id: asdf
4		4	

Test Scen Do not update any user's info if entered info is incorrect

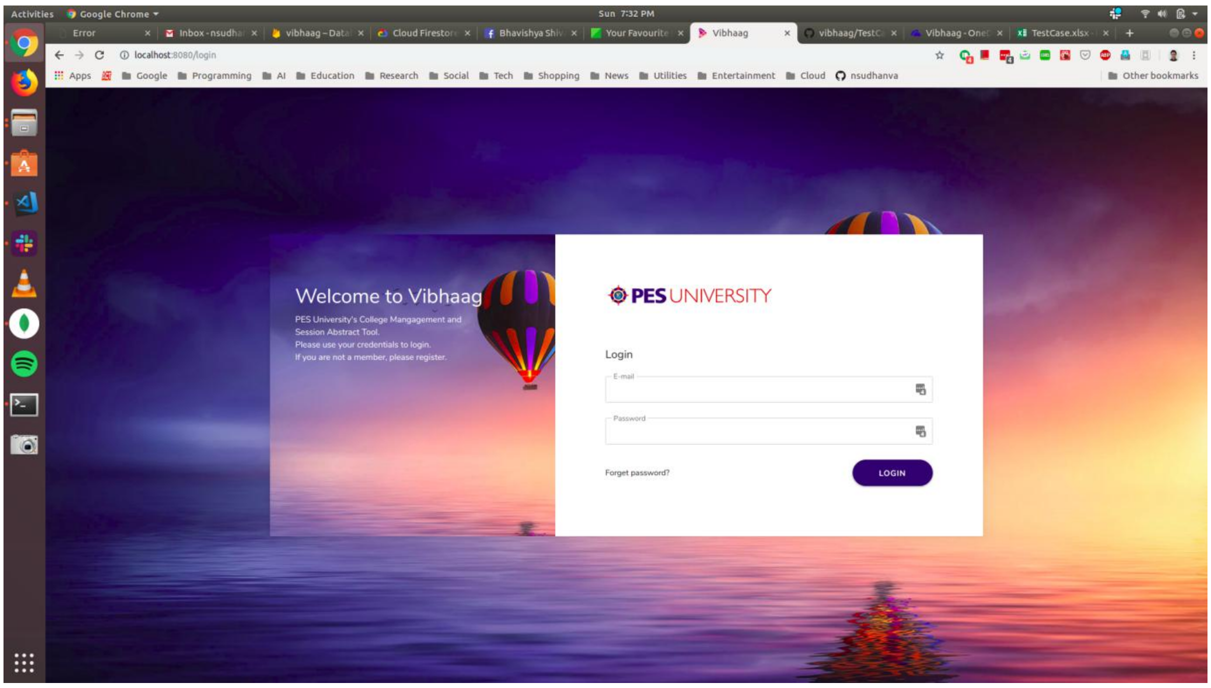
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	enter the things a user wants to update	no result expected	no result	Pass
2	make PUT request to localhost:3000/users/asdf	Do not update info. Return error message saying invalid details/user doesnt exist	does not update info. Returns error message saying invalid details/user does not exist.	Pass

SCREENSHOTS

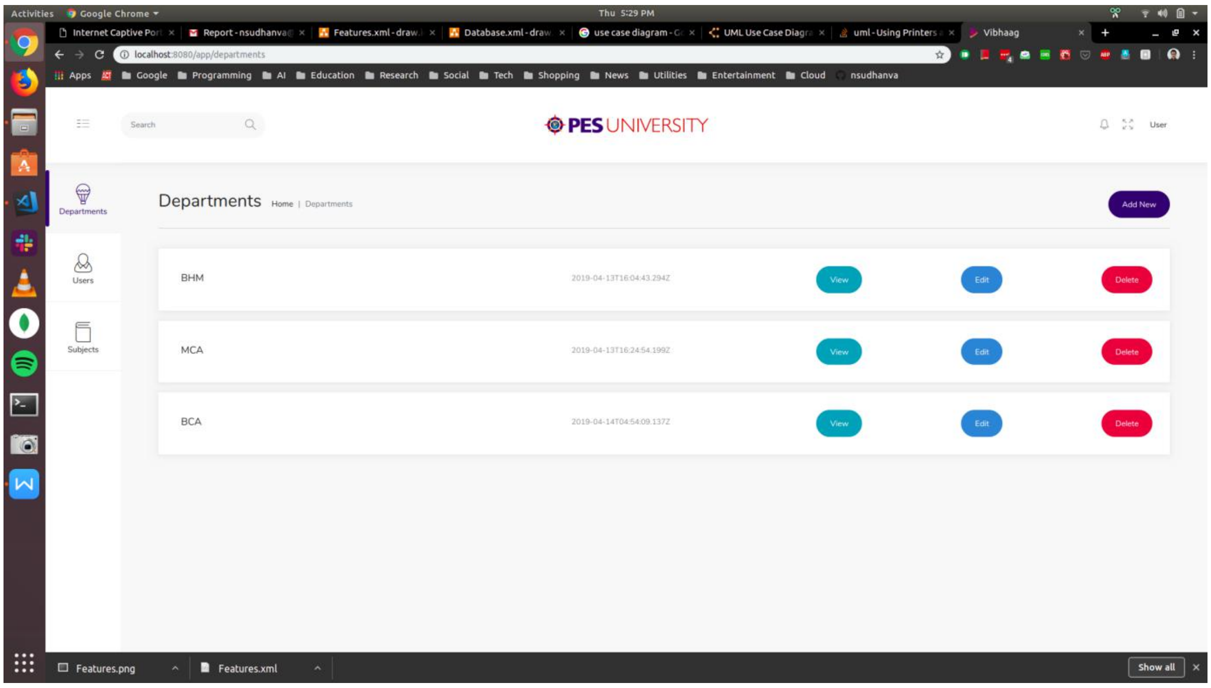
This is the screenshot of UI of user Registration



This is the screenshot of UI of user Login


localhost:8080/login
Welcome to Vibhaag
PES University's College Mangagement and Session Abstract Tool.
Please use your credentials to login.
If you are not a member, please register.
PES UNIVERSITY
Login
E-mail
Password
Forget password?
LOGIN

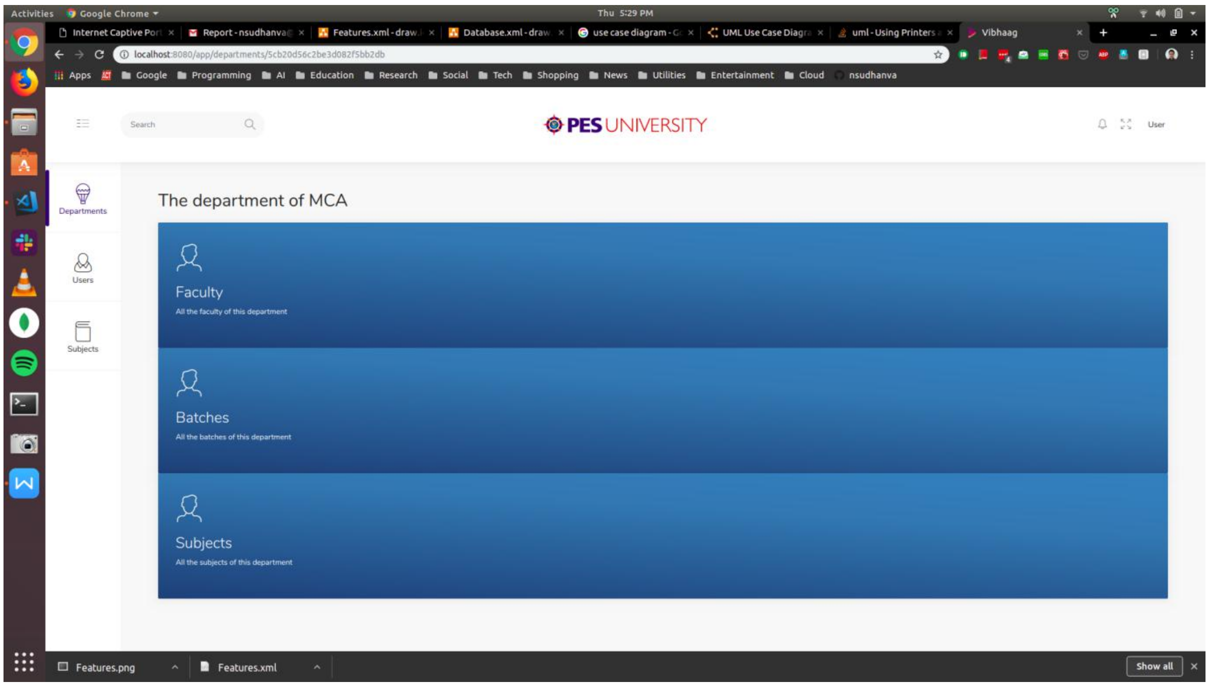
This is the screenshot of UI of department page



Departments Home | Departments

BHM	2019-04-13T16:04:43.294Z	View	Edit	Delete
MCA	2019-04-13T16:24:54.199Z	View	Edit	Delete
BCA	2019-04-14T04:54:09.137Z	View	Edit	Delete

This is the screenshot of UI of department home page



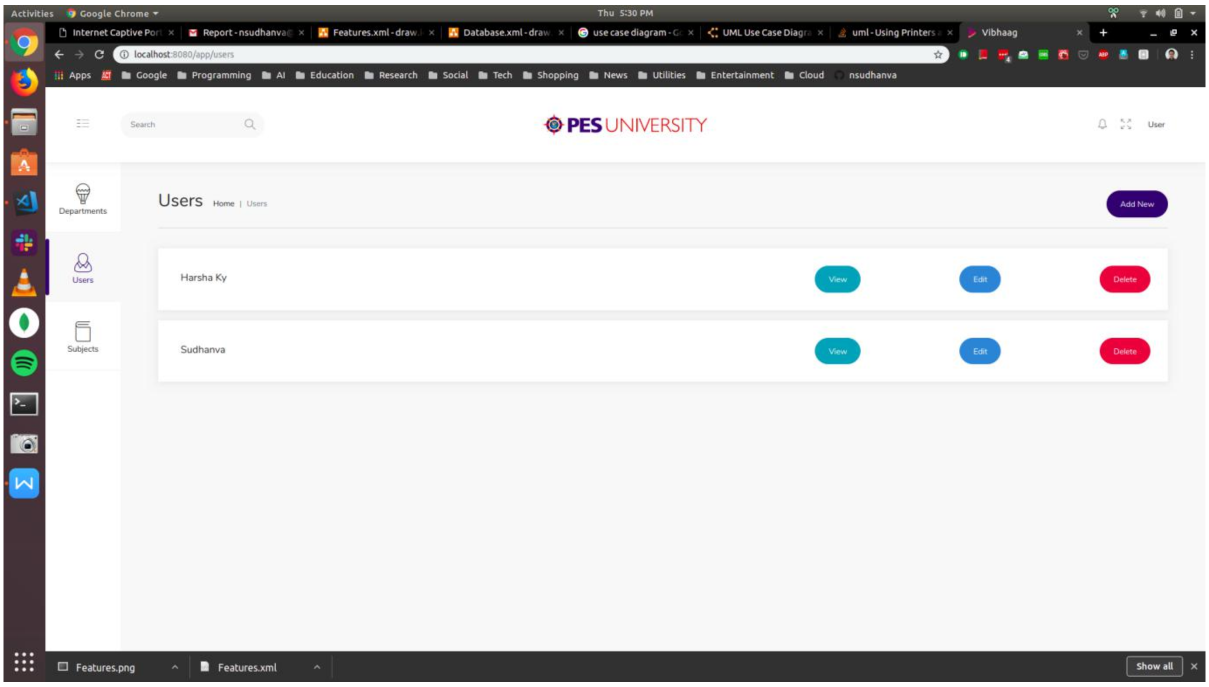
The department of MCA

Faculty
All the faculty of this department

Batches
All the batches of this department

Subjects
All the subjects of this department

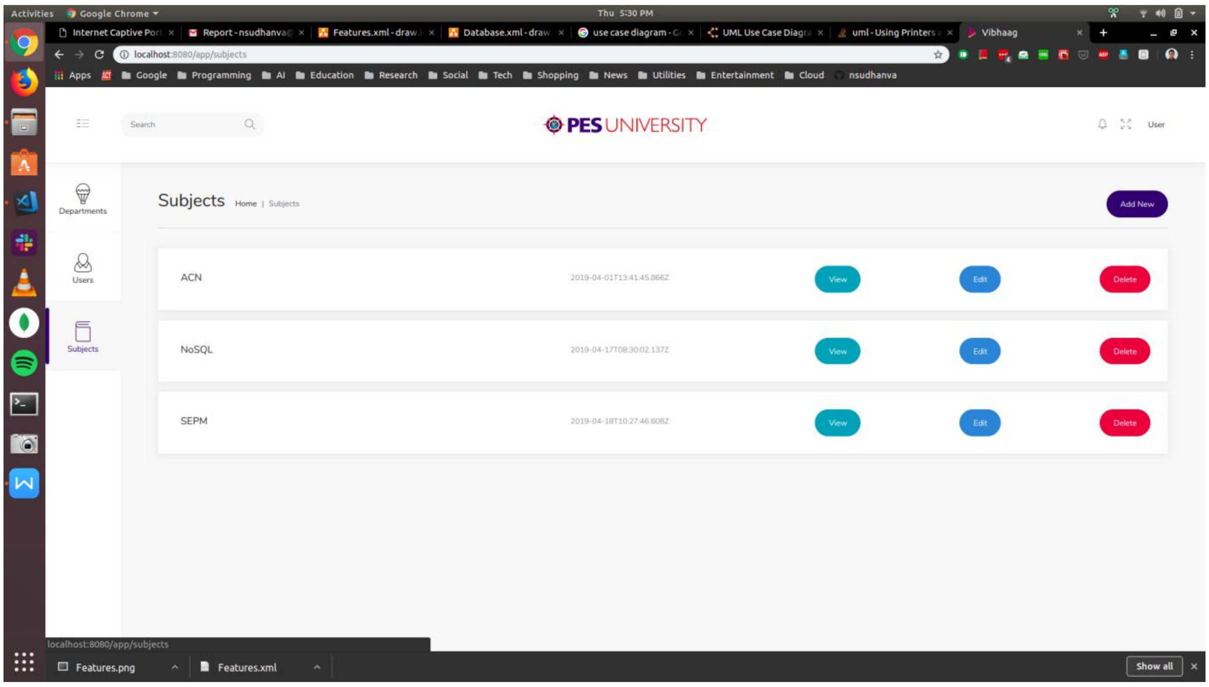
Screenshot of all users



Users Home | Users

Harsha Ky	View	Edit	Delete
Sudhanva	View	Edit	Delete

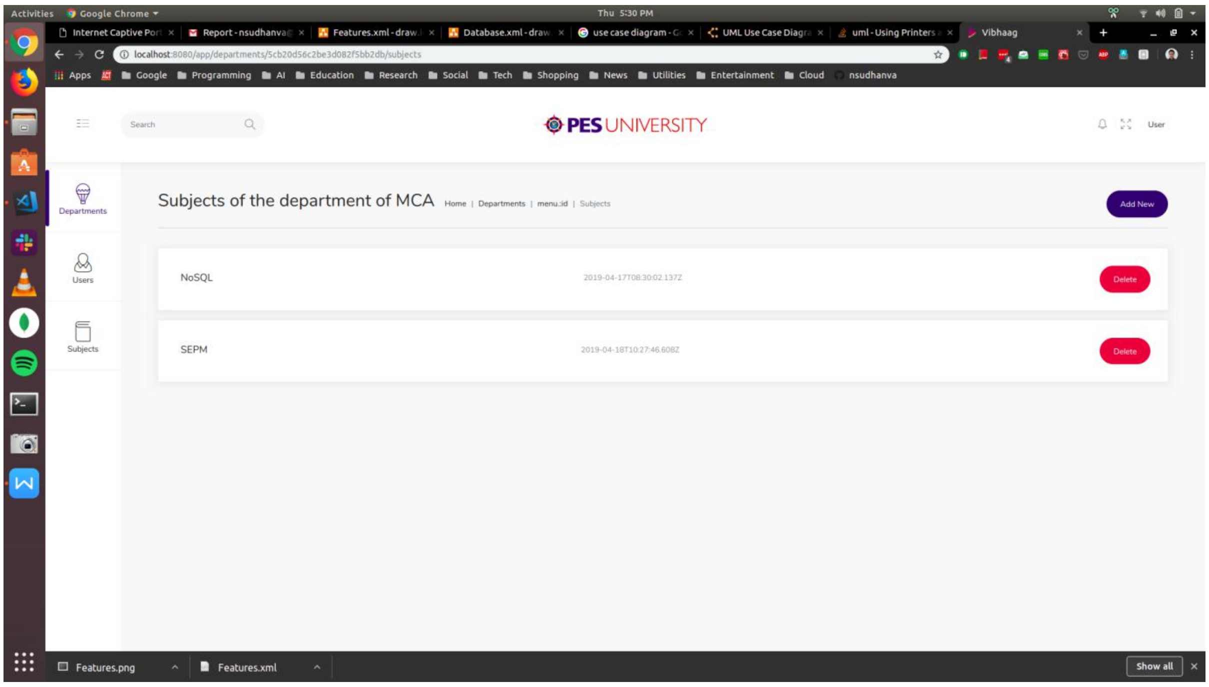
Screenshot of all subjects



Subjects Home | Subjects

ACN	2019-04-01T13:41:45.866Z	View	Edit	Delete
NoSQL	2019-04-17T08:30:02.137Z	View	Edit	Delete
SEPM	2019-04-18T10:27:46.608Z	View	Edit	Delete

Screenshot of all Subject of a particular department


localhost:8080/app/departments/5cb20d56c2be3d082f5bb2db/subjects
PES UNIVERSITY
Search
User
Departments
Users
Subjects
Home | Departments | menu.id | Subjects
Add New
Delete
Delete
Features.png
Features.xml
Show all

Subjects of the department of MCA

NoSQL 2019-04-17T08:30:02.137Z

SEPM 2019-04-18T10:27:46.608Z

CONCLUSION

- This application tries to solve a real life problem at an affordable cost. It takes the best features of few applications and embeds them in itself.
- This application is simple, user friendly and has a formal UI.
- The project is currently under the development phase where it is being tested for accuracy and reliability.
- It will help every enterprise in making their day to day processes easier and accountable.
- Further plan is to deploy the product in any enterprise organization which strive for quality.

FUTURE ENHANCEMENT

- Now the application records the actual event based on QR-Code scanned. In future, location based authentication, like co-ordinates of classroom, will be more efficient
- Now the application depends on user’s email and password for account validation. In addition to that device fingerprint authentication can be added.
- Implementation of smart time table scheduler, automatic assignment of faculty if there is shortage or emergencies

BIBLIOGRAPHY

- ReactJS documentation: https://reactjs.org/
- Redux documentation: https://redux.js.org/
- NodeJS documentation: https://nodejs.org/en/
- Firebase documentation: https://firebase.google.com
- Medium blog: https://medium.com
- FreeCodeCamp blog: https://freecodecamp.org