Vibhaag Web App

A college analytics, session monitoring and management tool

Guide:

Dr Veena S,
Chairperson,
Department of Computer Applications

By,

Sudhanva N
PES1201702260

Karthik D
PES1201702449

Harsha K Y
PES1201801839

Abstract

- The application consists of two components
- The web application which helps monitor and manage the activities of the college
- The mobile application for end users that acts as a companion and authentication tool
- It records the time, date and location of the incident by a QR Code and sends the data securely to the web application

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.
- **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).
- **TrackerPal** - Time and Attendance Software with Geofencing. Easily manage remote and field employee attendance. Employees punch in and out using their mobile phone.
- **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.

Introduction

- Custom time table for every batch and for every session
- Faculties can confirm the class conduction using mobile apps with timings
- Meaningful insights such as daily and weekly abstracts can be generated
- Admins have the ability to change the time table and re-allocate faculties as per requirement

Tools and Technologies

Core

- Backend:
  - Node.js v10.x, Express.js v8.x
- Frontend:
  - React.js v16.x, Redux v16.x
- Database:
  - MongoDB v4.x

Deployment

- Testing:
  - Mocha, Postman
- DevOps:
  - Github, Travis CI, AWS, Heroku
- Other Tools:
  - Draw.io, Balsamiq

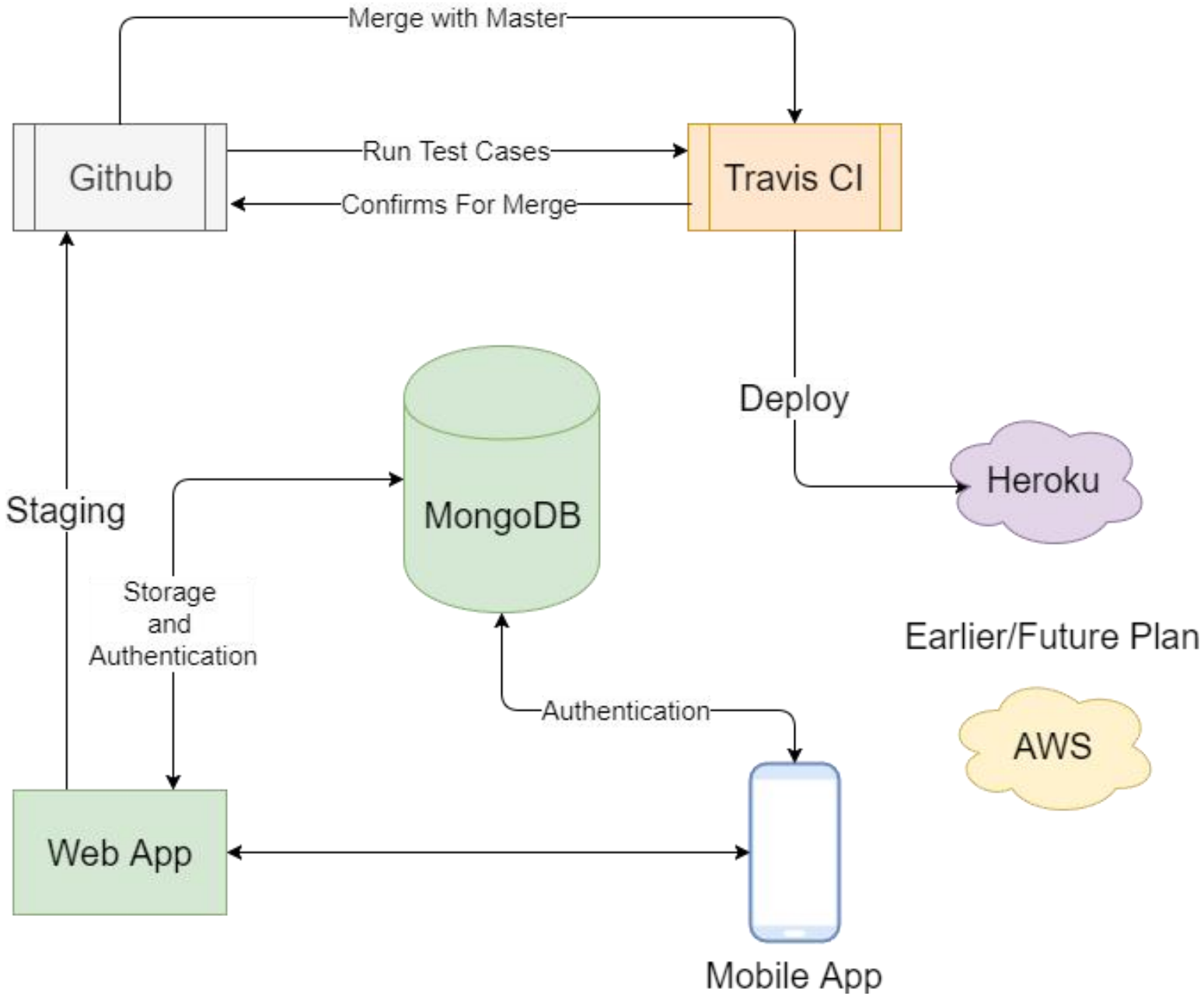
Functional Requirements

- User authorization and authentication with role management
- User should have restricted access, permission based creation of departments, faculty or any other entity
- Customized schedules for sessions for different user or group along with notifications
- Create departments, sessions, add faculty, add subjects

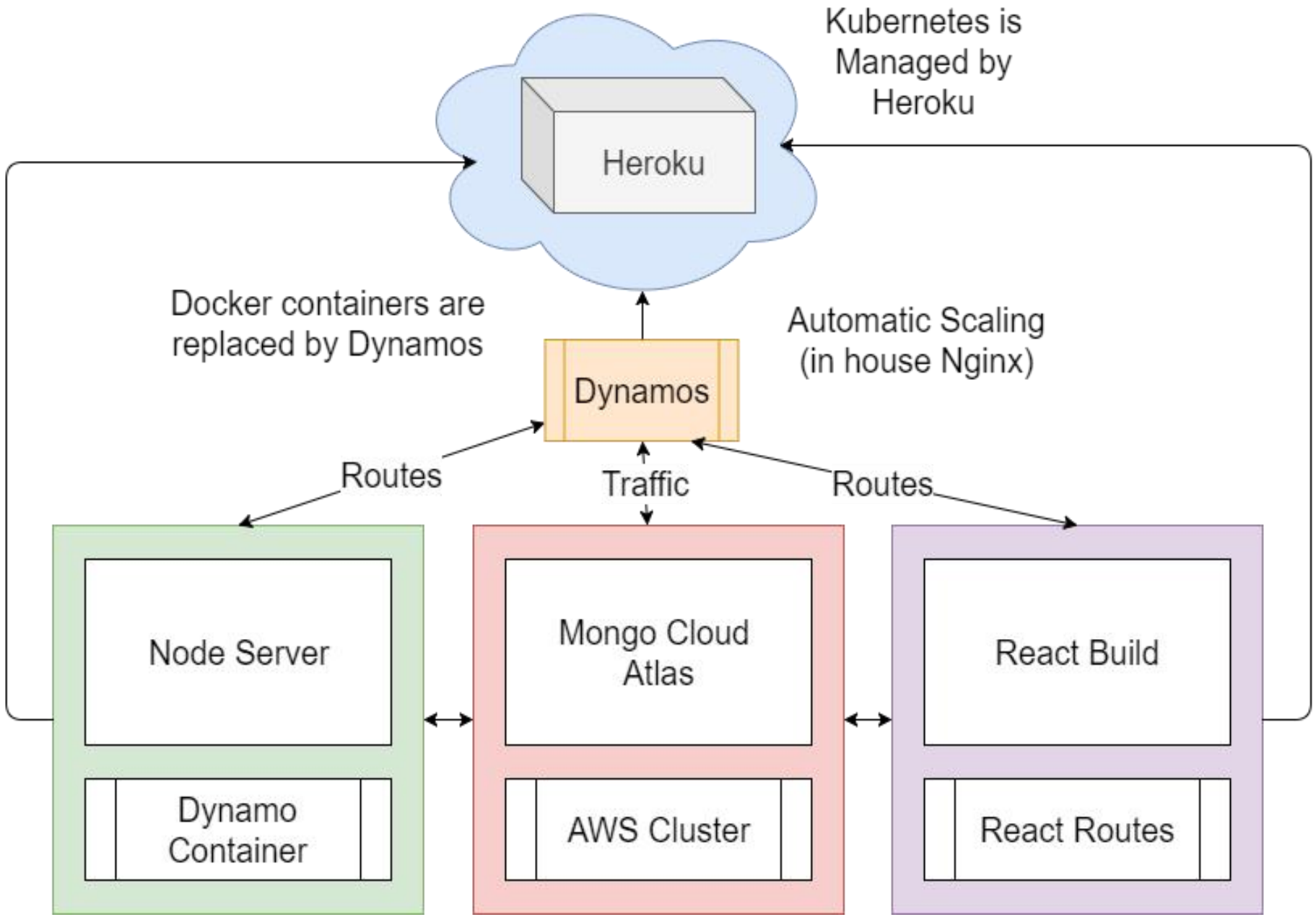
Non - Functional Requirements

- **Performance:** Application is build keeping scalability in mind meaning that is can serve lots of users without fail
- **Availibility:** Since it is an online application, it is available at all times as long the user has an active internet connection
- **Maintainability:** Version control with continuous integration and continuous delivery for seamless development operations
- **Data Integrity:** Data is store in multiple nodes as a backup in case of emergenices or recovery

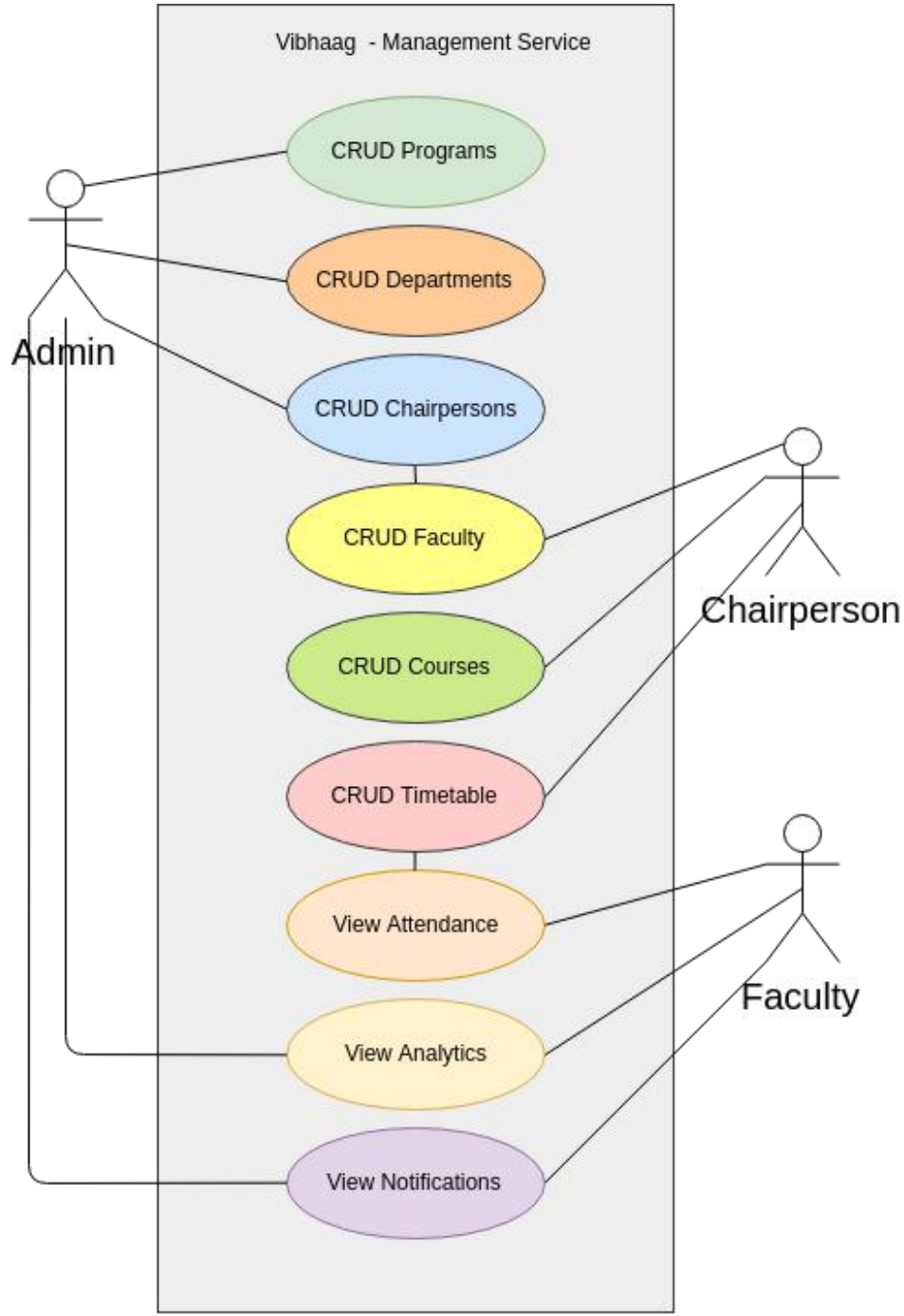
Process Flow



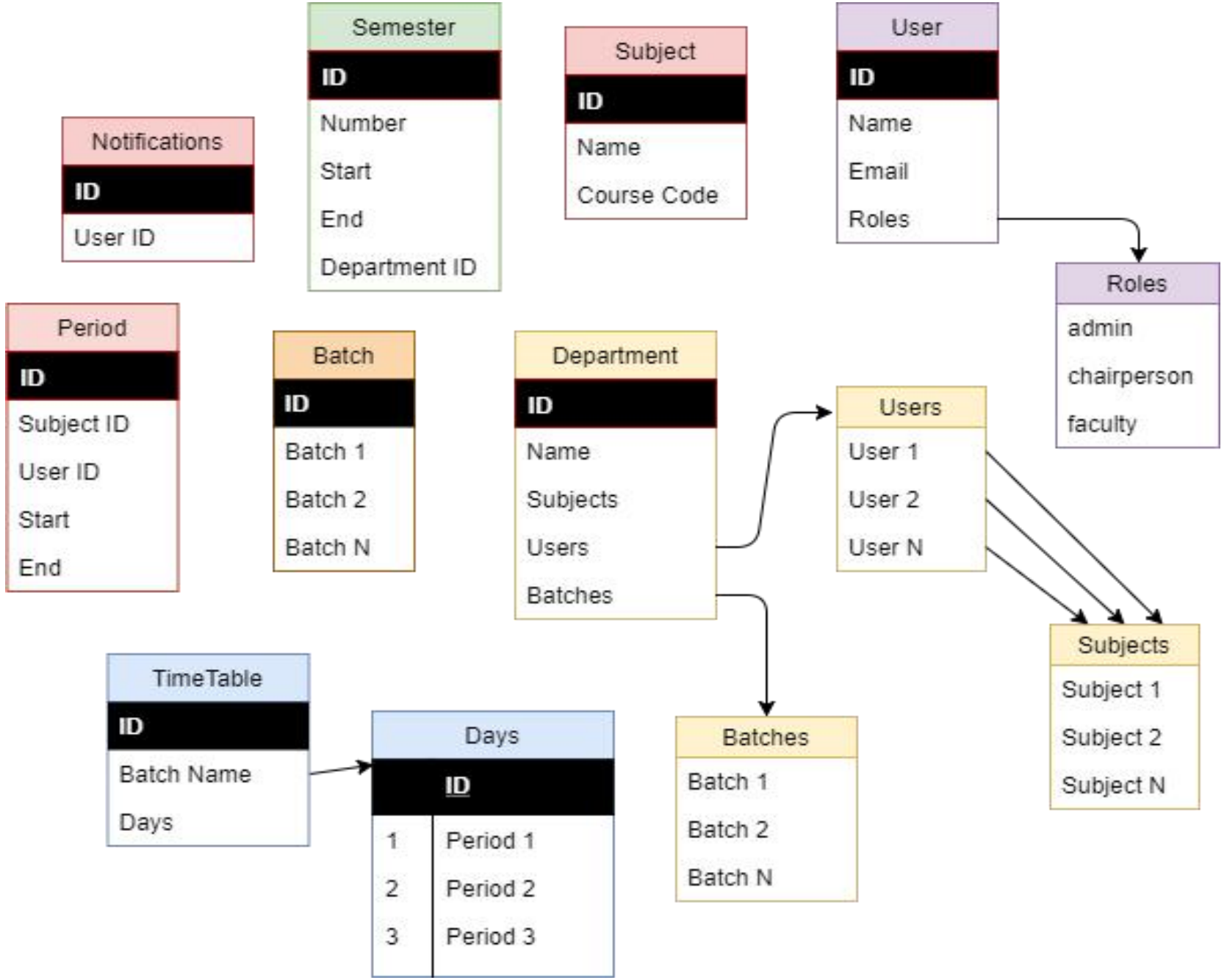
System Design



Use Cases


Vibhaag - Management Service
CRUD Programs
CRUD Departments
CRUD Chairpersons
CRUD Faculty
CRUD Courses
CRUD Timetable
View Attendance
View Analytics
View Notifications
Admin
Chairperson
Faculty

Database Structure



Test Cases

Test Case ID	1	Test Case Description	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	Pass

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

S #	Test Data
1	No data to be passed as it is a HTTP GET request
2	
3	
4	

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

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

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	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 u

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

S #	Prerequisites:
1	Access to Chrome Browser
2	Test server running on local machine
3	Postman opened to simulate HTTP requests
4	

S #	Test Data
1	id: asdfqwer
2	id: 5c9e68c9cdcaf64708842fqw
3	
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://localho	Should return error message saying invalid id	As Expected	Pass
2	Make a get request to localhost:3000/users/5c9e68c9cdcaf64708842fqw on Postman	Should return error message saying user doesnt exist	As Expected	Pass

Test Case ID	4	**Test Case Description**	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**	Pass

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

S #	Test Data
1	id: 5c9e68c9cdcaf64708842fad
2	
3	
4	

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

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	As Expected	Pass

Test Case ID	5	**Test Case Description**	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**	Pass

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

S #	Test Data
1	id: asdfqwer
2	id: 5c9e68c9cdcaf64708842fqw
3	
4	

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

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

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

S #	Test Data
1	name: Harsha Ky
2	email: harsha@gmail.com
3	password: userpassword
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	As Expected	Pass
2	Make a POST request to	Should create a new user if a user with same email doesn't	As Expected	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**	Pass

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

S #	Test Data
1	name: Harsha Ky
2	email: asdfqwer1234
3	s
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	As Expected	Pass
2	make a POST request to localhost:3000/users	should return error message saying wrong/invalid information	As Expected	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	Pass

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

S #	Test Data
1	email: harsha@gmail.com
2	password: userpassword
3	
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	As Expected	Pass
2	Make a POST request to	Should create an x-auth token and return back the token as header and save the	As Expected	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**	Pass

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

S #	Test Data
1	email: harsha@gmail.com
2	password: asfwrq21r4
3	
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	As Expected	Pass
2	Make POST request to localhost:3000/users/login	Should return error message saying entered details are incorrect and should	As Expected	Pass

Test Case ID	10	Test Case Description	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	Pass

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

S #	Test Data
1	email: harsha@gmail.com
2	name: Harsha Ky
3	id: 5c9e68c9cdcaf64708842fad
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	As Expected	Pass
2	Make a PUT request to	Should update the info of the user to whom the id belongs and return back the	As Expected	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**	Pass

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

S #	Test Data
1	email: harsha@gmail.com
2	name: asdf
3	id: asdf
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	As Expected	Pass
2	make PUT request to localhost:3000/users/asdf	Do not update info. Return error message saying invalid details/user doesnt	As Expected	Pass

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 inany enterprise organization which strive for quality.

Bibiliography

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

Future Enhancement

- Now the application records the actual event based on QR-Code scanner. In future, location based authentication, like co-ordinates of classroom, will be more efficient

- 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

Demo