

2

# PROJECT REPORT

**ON**

**DELIGHT DASHBOARD**

**BY**

### PARTH AGGARWAL 2018A8PS0041G

### PRIYESH KANT 2018A8PS0467G

**Prepared in partial fulfillment of the Practice School – II Course**

### at

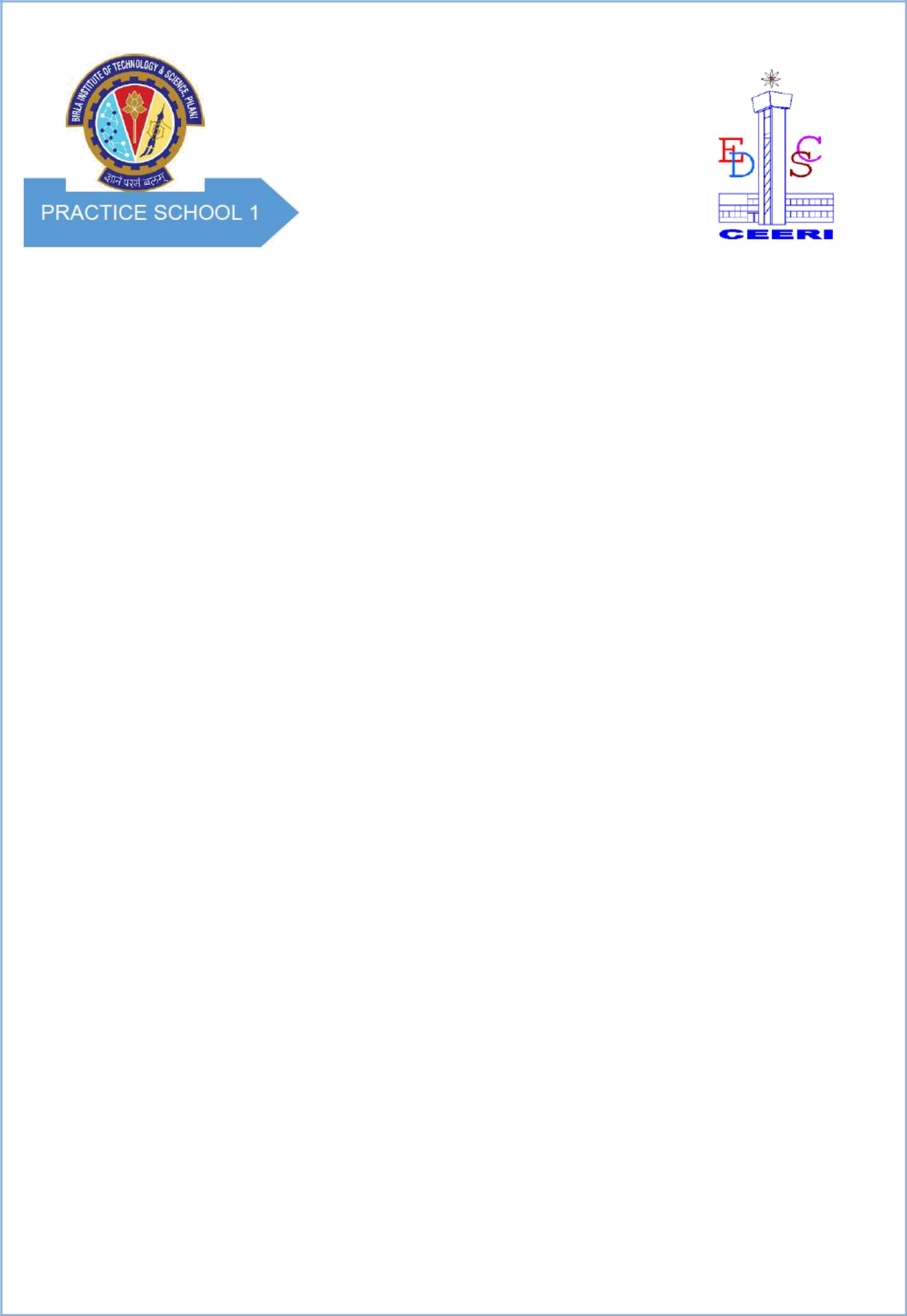


**Classplus**

**A Practice School-II station of**

### BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

**December, 2021**

**1 |** P a g e

**A REPORT ON**

**DELIGHT DASHBOARD**

# BY

### Name of the ID.No. Discipline Student

**PARTH AGGARWAL 2018A8PS0041G Electronics & Instrumentation**

### PRIYESH KANT 2018A8PS0467G Electronics & Instrumentation

**Prepared in partial fulfillment of the Practice School-II Course**

*AT*

**Classplus**

A Practice School- II station of

### BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

**December, 2021**

**ACKNOWLEDGEMENT**

We would like to thank everyone for providing me this opportunity, to learn and utilize my knowledge throughout the course of this project. We are extremely thankful to everyone for providing me their valuable insights, proper guidance and timely advice. We are sincerely grateful to them for helping me through any difficulties we have been facing in this project.

We would also like to thank **Multiverse Team at Classplus** for motivating us and providing us this opportunity to work at this esteemed institute. We would also like to thank my PS-Faculty Prof. **Sugata Ghosal**for their co-operation. We would also like to extend my heartfelt gratitude to **Devansh Arora** for providing us his valuable time and giving us this great opportunity of working under him on such an interesting topic.

We would especially like to thank Practice School Division for their constant support, maintaining discipline and helping us make the most of our time during PS-2.

### BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

**Practice School Division**

**Station** ……… Classplus. ………. **Centre** ..................Noida… **Duration**....................153 days………………… **Date of Start**.........July 5, 2021……

**Date of Submission** ..................…………December 5, 2021……………………………………

**Title of the Project**: DELIGHT DASHBOARD

### Name(s) Discipline(s)/of the student(s) ID No.

PARTH AGGARWAL Electrical and Instrumentation 2018A8PS0041G

PRIYESH KANT Electrical and Instrumentation 2018A8PS0467G

### Name(s) and designation(s) of the expert(s):

Mr. Devansh Arora, Lead Engineer at Classplus

**Name(s) of the PS Faculty:** SUGATA GHOSAL

**Project Areas:** Web Development, Backend Development

Signature of Students Signature of PS Faculty

Date: 05/12/2021 Date:05/12/2021

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| ***i.*** | ***Cover Page*** | ***1*** |
| ***ii.*** | ***Title Page*** | ***2*** |
| ***iii.*** | ***Acknowledgements*** | **3** |
| ***iv.*** | ***STATION DETAILS*** | **4** |
| **1.** | **Introduction** | **6** |
| **2.** | **About the Company** | **7** |
| **3.** | **PROJECT-DELIGHT DASHBOARD** | **8** |
| **4.** | **Technology Stack** | **9** |
|  |  |  |
| **5.** | **Major Tasks** | **17** |
|  |  |  |
| **6.** | **MVC Architecture** | **18** |
| 1. **Cron scheduling 19**   **8. Do It Yourself Application 20**   * 1. **9. Docker 21**   2. **10. User Management System 22**   3. **11. MicroServices Architecture 23**   **12. Major Challenges and Problems and References 25** | | |

# INTRODUCTION

This report’s primary objective is to elaborate on our project – Delight Dashboard and the desired features made using different technology stack.

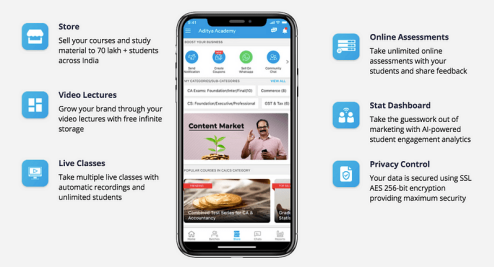
The Multiverse is a CRM (Customer Relationship Management) system that strategies techniques, tools, and technologies to develop, retain and acquire customers.

Its major goals are to build advance automated tools which help internal

sales teams, accounts teams, and many more to manage their tasks and responsibilities which in turn make any interaction with the customer go smoothly and efficiently.

Another major goal of the multiverse is to generate more business for the company by identifying relevant customers and help in the overall business growth of the company.  
  
The domain of this system is very vast and end to end, which varies from identifying potential leads to managing their entire account with smart automated tools.

**ABOUT THE COMPANY**



Classplus works on B2B business model as it helps the educators

to bring their offline businesses to online mode. It offers educators

to take multiple live classes & teach unlimited students across the country without worrying about storage and video quality.

Not only that, educators can share their notes, post their pre-recorded lectures, collect fees, track their student's performance

and growth and much more. This gives an edge to improve their online teaching experience.

Our work is to build a microservice platform that strategies techniques, tools, and technologies to develop, retain and acquire customers and make the business of company profitable.

**PROJECT-DELIGHT DASHBOARD**

Our Work is to build a microservice platform called Multiverse.

The Multiverse is a CRM (Customer Relationship Management) system that strategies techniques, tools, and technologies to develop, retain and acquire customers.

Its major goals are to build advance automated tools which help internal sales teams, accounts teams, and many more to manage their tasks and responsibilities which in turn make any interaction with the customer go smoothly and efficiently.

Another major goal of the multiverse is to generate more business

for the company by identifying relevant customers and help in the overall business growth of the company.

The domain of this system is very vast and end to end, which

varies from identifying potential leads to managing their entire account with smart automated tools.

# TECHNOLOGY STACK

# JAVASCRIPT

# NODE JS

# MYSQL

# AURORA DATABASE

# REDIS

# SWAGGER UI

# TYPE ORM

# EXPRESS

# AWS SQS

# MOCHA,CHAI AND SINON

# TECH STACK EXPLAINED IN BRIEF

# JAVASCRIPT

# JavaScript Core Language Learning Path | Pluralsight

# JavaScript is commonly used for creating web pages. It allows us to add dynamic behavior to the webpage and

# add special effects to the webpage. On websites, it is

# mainly used for validation purposes. JavaScript helps us

# to execute complex actions and also enables the

# interaction of websites with visitors.

# NODE JS

# What Is Node.JS and What Is It Used for? – CloudSavvy IT

# It is used for server-side programming, and primarily

# deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based

# architectures in mind. Every browser has its own version

# of a JS engine, and node.

# MYSQL

# Learn SQL: What You Should Know About SQL Before Getting Started in 2021 | Berkeley Coding Boot Camp

# SQL (Structured Query Language) is a standardized programming language that's used to manage [relational databases](https://searchdatamanagement.techtarget.com/definition/relational-database) and perform various operations on the data in them.

# The [uses of SQL](https://searchsqlserver.techtarget.com/tip/SQL-language-crash-course-just-enough-to-be-dangerous) include modifying database table and index structures; adding, updating and deleting rows of data; and

# retrieving subsets of information from within a database for transaction processing and analytics applications. Queries and

# other SQL operations take the form of commands written as

# statements -- commonly used SQL statements include select, add, insert, update, delete, create, alter and truncate.

# AMAZON AURORA

# Capturing Data Changes in Amazon Aurora Using AWS Lambda | AWS Database Blog

* Aurora automatically allocates database storage space in 10-gigabyte increments, as needed, up to a maximum of 128 terabytes.
* Aurora offers automatic, six-way replication of those chunks across three Availability Zones for improved availability and fault-tolerance.
* Aurora provides users with performance metrics, such as

query throughput and latency. It provides fast database

cloning.

* Aurora Multi-Master allows creation of multiple read-write

instances in an Aurora database across multiple Availability

Zones, which enables uptime-sensitive applications to

achieve continuous write availability through instance failure.

# REDIS

# What is Redis? How to Install and Basic Commands

# Redis offers purpose-built in-memory data structures and operators to manage real-time geospatial data at scale and speed. Commands such as GEOADD, GEODIST, GEORADIUS, and GEORADIUSBYMEMBER to store, process, and analyze geospatial data in real-time make geospatial easy and fast with Redis.

# SWAGGER UI

# ASP.NET Core Swagger UI Authorization using IdentityServer4

# Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API’s resources without

# having any of the implementation logic in place. It’s automatically generated from your OpenAPI (formerly known as Swagger) Specification, with the

# visual documentation making it easy for back end implementation and client side consumption. Use Swagger UI to generate interactive API

# documentation that lets your users try out the API calls directly in

# the browser. Use the spec to connect API-related tools to your API.

# TYPE ORM

# TypeORM MongoDB Review. I recently started using TypeORM in a… | by Eliezer Steinbock | Medium

TypeORM framework is an **Object Relational Mapping (ORM)**

framework. In general, **Object** part refers to the domain / model in

your application, **Relational** part refers to the relationship between

tables in Relational Database Management System (e.g. Oracle,

MySQL, MS-SQL, PostgreSQL, etc.) and finally the **Mapping** part

refers to the act of bridging the model and our tables.

ORM is a type of tool that maps entities with database tables. ORM

provides simplified development process by automating object-to-table

and table-to-object conversion. Once you can write your data model in

one place, it becomes easier to update, maintain, and reuse the code.

# EXPRESS.JS

# Express.js Tutorial for Beginners | Learn Express Fundamentals | Edureka

ExpressJS is a web application framework that provides you with

a simple API to build websites, web apps and back ends. With ExpressJS, you need not worry about low level protocols,

processes, etc.Express provides a minimal interface to build our applications. It provides us with the tools that are required to build our app. It is flexible as there are numerous modules available on npm, which can be directly plugged into Express. Express. js is a

free and open-source web application framework for Node. js. It is used for designing and building web applications quickly and easily.

# AWS SQS

# 

Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components.There

are three main parts in a distributed messaging system: the

components of your distributed system, your queue (distributed

on Amazon SQS servers), and the messages in the queue.

In the following scenario, your system has several *producers* (components that send messages to the queue) and *consumers* (components that receive messages from the queue). The queue (which holds messages A through E) redundantly stores the messages across multiple Amazon SQS servers.

**MOCHA,CHAI AND SINON**

****

Tests help document the core features of an application. Properly written tests ensure that new features do not introduce changes

that break the application.An engineer maintaining a codebase might not necessarily be the same engineer that wrote the initial code. If the code is properly tested another engineer can

confidently add new code or modify existing code with the expectation that the new changes do not break other features or,

at the very least, do not cause side effects to other features.

**Mocha** is a feature-rich JavaScript test framework running on

Node.js and in the browser. It encapsulates tests in test suites (describe-block) and test cases (it-block).It has features like

browser support,simple async support including promises,test coverage reporting,async test timeout support.

To make equality checks or compare expected results against

actual results we can use Node.js built-in assertion module. However, when an error occurs the test cases will still pass. So Mocha recommends using other assertion libraries and for this project, we used **Chai**.It exposes three assertion interfaces:

expect(), assert() and should(). Any of them can be used for assertions.

Often, the method that is being tested is required to interact with

or call other external methods. Therefore you need a utility to spy,stub,or mock those external methods. This is exactly what

**Sinon** does for us.Stubs, mocks, and spies make tests more

robust and less prone to breakage should dependent codes

evolve or have their internals modified.A spy is a fake function

that keeps track of arguments, returns value, the value of this and exception is thrown (if any) for all its calls.A stub is a spy with predetermined behavior.We can use a stub to take a

predetermined action, like throwing an exception ,provide a predetermined response,prevent a specific method from being

called directly (especially when it triggers undesired behaviors

like HTTP requests).A mock is a fake function (like a spy) with

pre-programmed behavior (like a stub) as well as

pre-programmed expectations.We can use a mock to verify the contract between the code under test and the external methods

that it calls,verify that an external method is called the correct number of times,verify an external method is called with the

correct parameters.

**MAJOR TASKS**

Engineering tasks which we built include a robust backend

platform which includes making of Application Programming Interfaces (APIs) which will be used by Front End Development

Team to make the product work dynamically and in a high end manner.

* We made various APIs of both White Label for the Multiverse (White Label is an internal sub project of Multiverse which

gives a platform to manage all organizations that are

associated with the ClassPlus)

* APIs in the multiverse-ts codebas which manages all

influencers of store team which includes CRUD (create, read, update and delete) along with other functionalities.

* Also we made wrappers for the multiverse team which is a

way of refracting code for better functioning of the product.

* Helped the multiverse team to upgrade their code for crons

to follow MVC (Model, View, Controller) Architecture

which previously they were not following.

* We made a script as well in python and node.js which fetches the details of users and tutors which will be used by internal team to track record of customers.

# MVC FRAMEWORK

The Model-View-Controller (MVC) is an architectural pattern

that separates an application into three main logical components:

the model, the view, and the controller. Each of these components

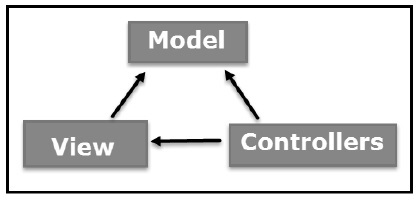
are built to handle specific development aspects of an

application.

MVC is one of the most frequently used industry-standard

web development frameworks to create scalable and

extensible projects.



### Model

The Model component corresponds to all the data-related logic that the user works with.

This can represent either the data that is being transferred between the View and

Controller components or any other business logic-related data. For example, a Customer

object will retrieve the customer information from the database, manipulate it and update it

data back to the database or use it to render data.

### View

The View component is used for all the UI logic of the application. For example, the Customer

view will include all the UI components such as text boxes, dropdowns, etc. that the final

user interacts with.

### Controller

Controllers act as an interface between Model and View components to

process all the business logic and incoming requests, manipulate data using

the Model component and interact with the Views to render the final

output. For example, the Customer controller will handle all the

interactions and inputs from the Customer View and update the database

using the Customer Model. The same controller will be used to view

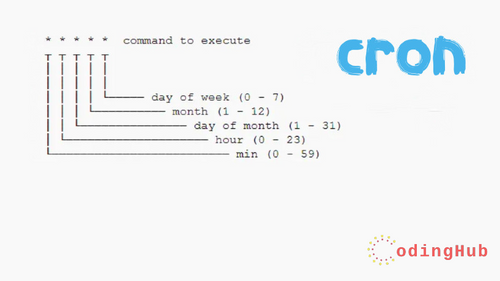
the Customer data.

## **CRON SCHEDULING**

No developer wants to spend all their time on tedious tasks such as system maintenance and administration, daily database backup, and downloading files and emails at regular intervals. You’d much rather focus on productive tasks instead of keeping track of when these bothersome chores need to get done. That’s where task scheduling comes in.Task scheduling enables you to schedule arbitrary code (methods/functions) and commands to be executed at a fixed date

and time, at recurring intervals, or once after a specified interval. In Linux operating systems, task scheduling is often handled by utility services such as cron at the OS level. For Node.js apps, cron-like functionalities are implemented using packages such as node-cron, which bills itself as a “tiny task scheduler in pure JavaScript for NodeJs.”

The actions of cron are driven by a crontab (cron table) file, a configuration file that contains instructions to the cron daemon. The node-cron module allows you to schedule tasks in Node using the full crontab syntax.A crontab syntax looks like this:

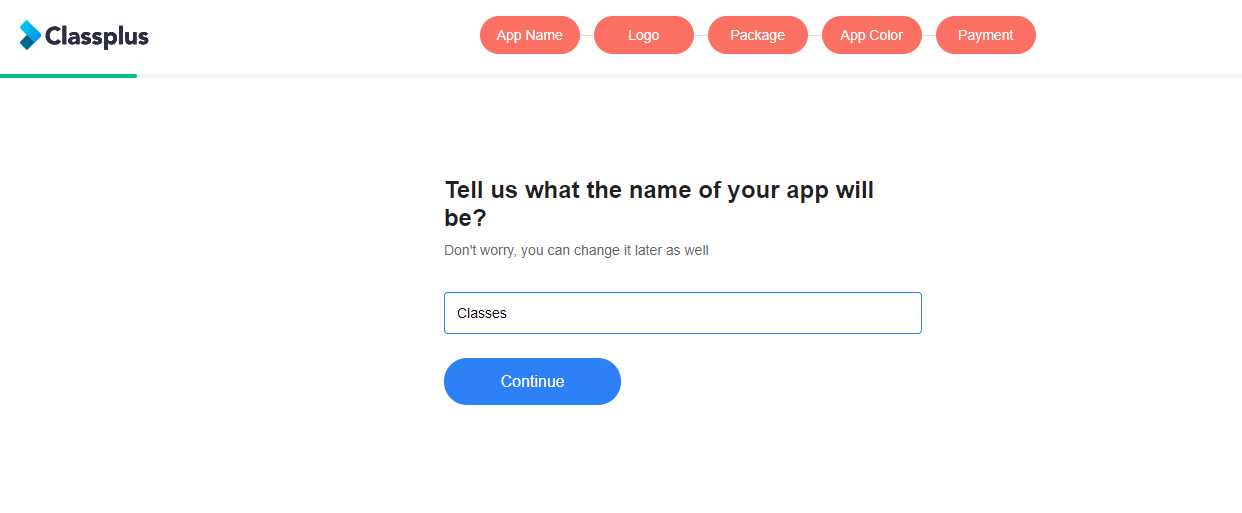


**DO IT YOURSELF APPLICATION**

An app that lets users/customers to create their own customized

app reducing their interaction with intermediaries and the time consumed in the process.Based on Node.js,Express.js,MySQL

tech stack and uses Typescript as the programming language.



**DOCKER**



[This Photo](https://manas.tech/blog/2015/12/15/logging-for-rails-apps-in-docker/) by Unknown Author is licensed under [CC BY-NC-ND](https://creativecommons.org/licenses/by-nc-nd/3.0/)

Docker is a set of platform as a service (PaaS) products that use

OS-level virtualization to deliver software in packages called containers.Containers are isolated from one another and bundle

their own software, libraries and configuration files; they can communicate with each other through well-defined channels. Because all of the containers share the services of a single

operating system kernel, they use fewer resources than virtual machines.The service has both free and premium tiers. The

software that hosts the containers is called Docker Engine.It was

first started in 2013 and is developed by Docker, Inc. Docker

works by providing a standard way to run your code. It is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server

hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides simple

commands you can use to build, start, or stop containers.

### The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests

and manages Docker objects such as images, containers,

networks, and volumes. A daemon can also communicate with

other daemons to manage Docker services.

### The Docker client

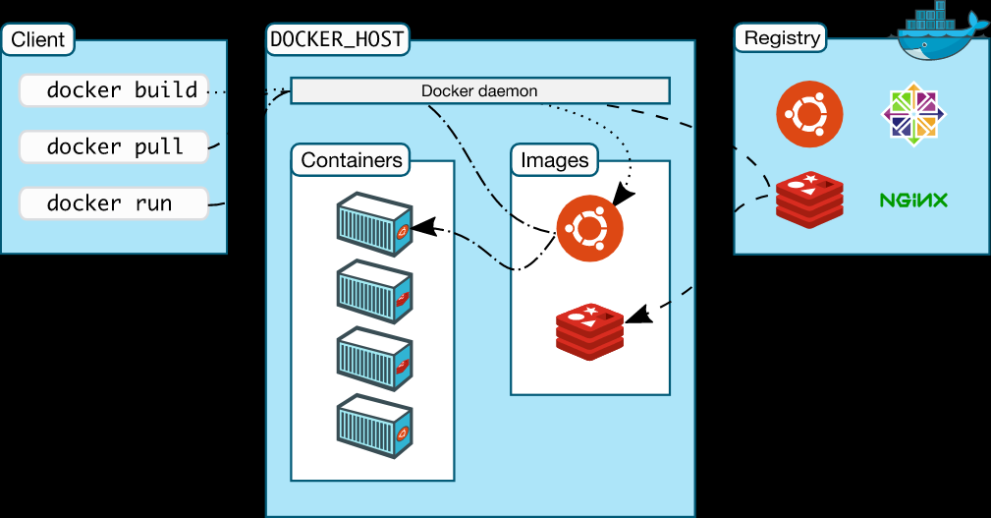
The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as

docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

#### Images

An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image,

with some additional customization.



#### Containers

A container is a runnable instance of an image. You can create,

start, stop, move, or delete a container using the Docker API or

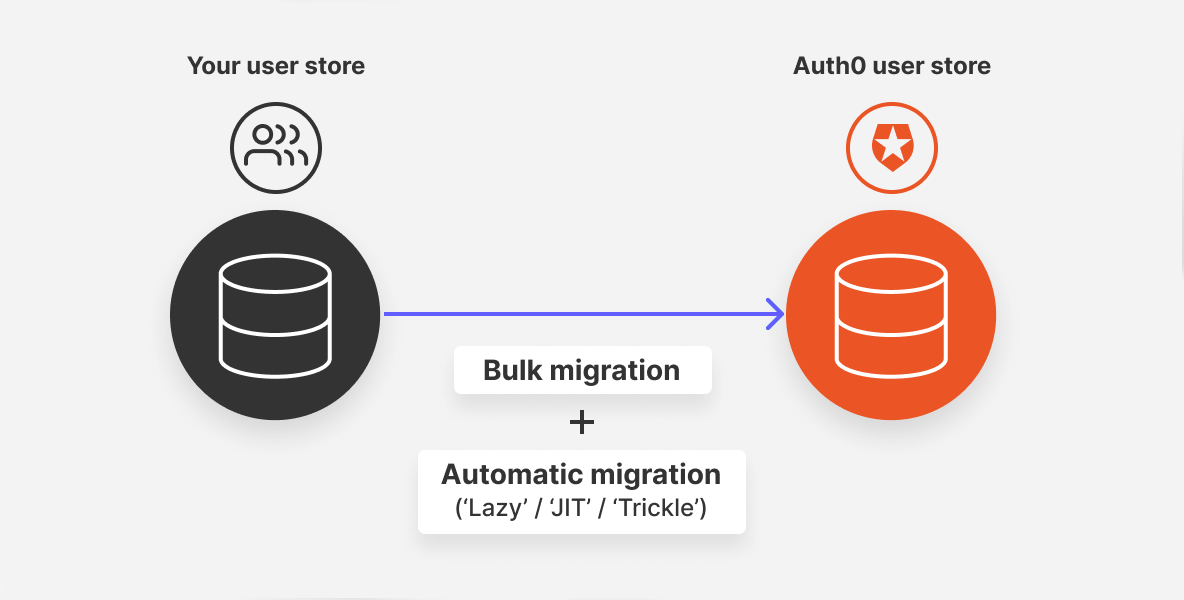
CLI.

**USER MANAGEMENT SYSTEM**

It create a delightful experience for company and its customers across the whole journey - from sign-up to purchase.

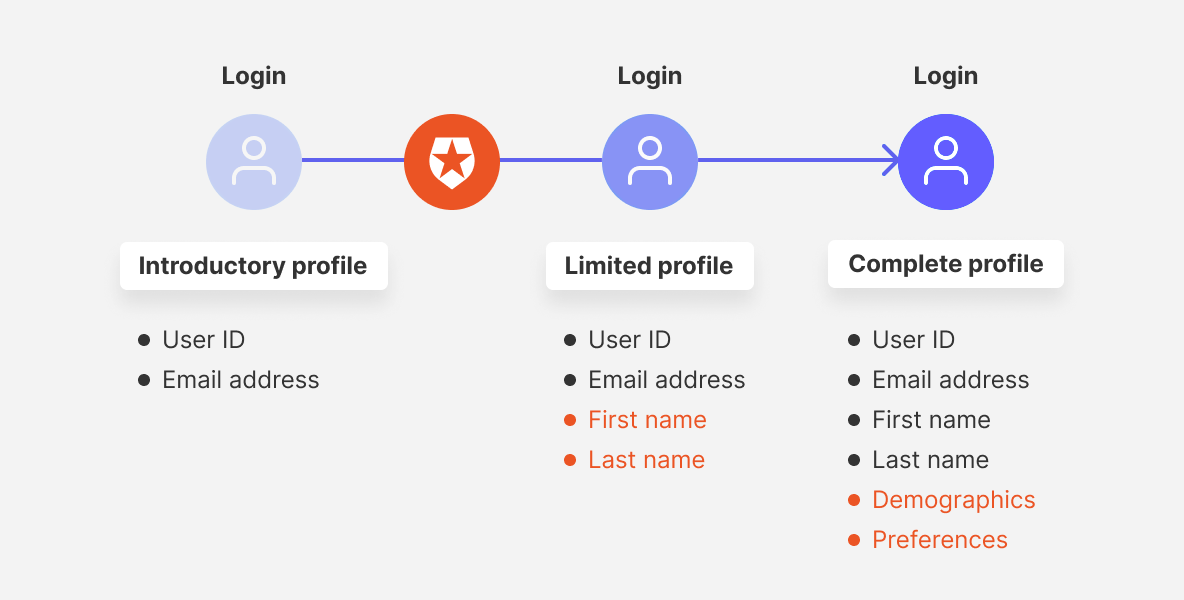
# User Migration

Migrating users can fill many application developers with dread. Auth0 provides multiple options to migrate them all at once, or gradually as they log-in. We make it easy for you, with zero disruption for your users.



# User Profiles

Get deeper insights into your customers’ wants and needs in a simple and easy-to-use management interface.



# User Connections

Deliver frictionless registration and login experiences. Enable

users to create accounts, use social providers, or, for your

enterprise customers, to federate easily with their identity provider. Give your user the freedom to choose from one or more of them

with account linking.

# MICROSERVICES ARCHITECTURE

# 

# A microservices architecture consists of a collection of small,

# autonomous services. Each service is self-contained and should implement a single business capability within a bounded context.

# A bounded context is a natural division within a business and provides an explicit boundary within which a domain model exists.

# Microservices are small, independent, and loosely coupled. A

# single small team of developers can write and maintain a service.

# Each service is a separate codebase, which can be managed by

# a small development team.Services can be deployed

# independently. A team can update an existing service without rebuilding and redeploying the entire application.Services are responsible for persisting their own data or external state. This

# differs from the traditional model, where a separate data layer handles data persistence.Services communicate with each other

# by using well-defined APIs. Internal implementation details of

# each service are hidden from other services.Supports polyglot programming. For example, services don't need to share the

# same technology stack, libraries, or frameworks.

**MAIN CHALLENGES AND PROBLEMS**

The main challenges we faced include understanding how to work

in teams collaborating with other teammates and writing

production level code.The technicalities include understanding of database of company and getting familiar with the structure of

their code particularly MVC(Model View Controller) and also we

had to revise my Javascript,OOP and DBMS concepts which

was vastly used by them in their code.

But detailed guidance by our mentor was extremely helpful in

every possible way to make us understand everything and help

us contribute to the project.

**References**

**MVC Framework**

<https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>

**Cron Scheduling in JS**

<https://enlear.academy/task-scheduling-with-node-js-20a3d46a86d6>

**Amazon SQS**

<https://docs.aws.amazon.com/sqs/index.html>

**Unit Testing in JS**

<https://howchoo.com/javascript/javascript-unit-testing-mochajs-chai-sinon>