# Industrial

# Training on

Web and App Development

SUBMITTED

BY

Name: Darshil Shah Section: D

Reg. No:180905298 E-mail: shah.darshil030101@gmail.com

Roll No :42 Phone: +91 9619751973

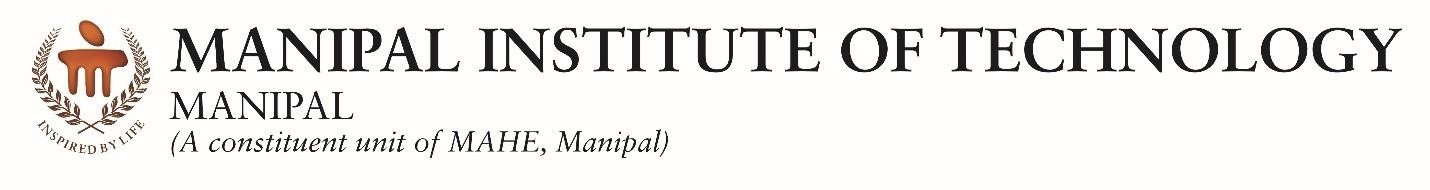
*Under the Guidance of:*

### 

SHIVAM AGARWAL

Director

Halara Labs Pvt. Ltd.



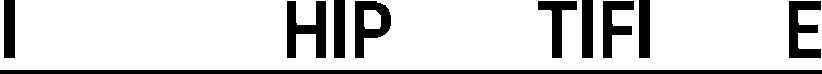
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

November 2020

HALARA LABS



CIN NO: U72900RJ2020PTC069333



Date: 20/Nov/2020

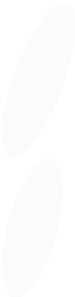
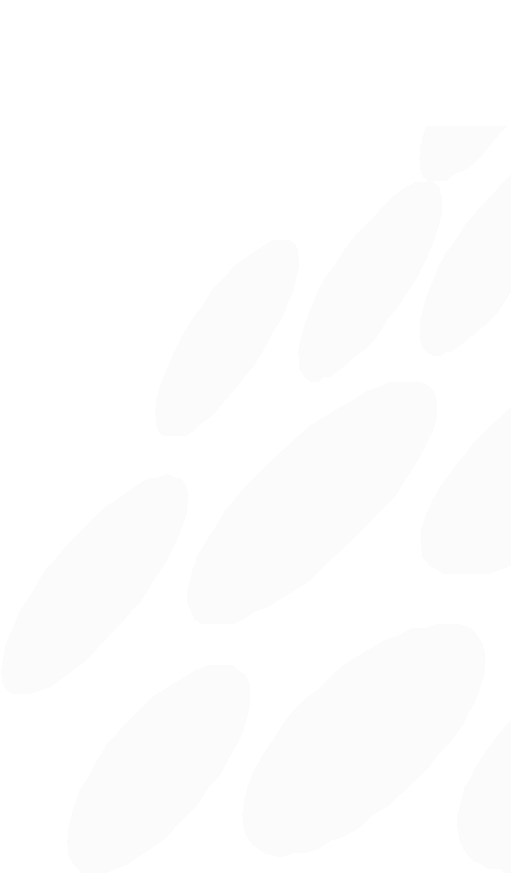
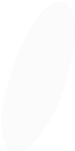
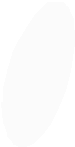
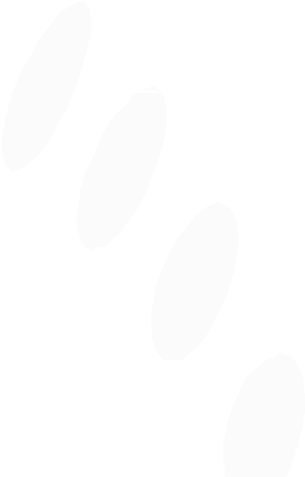
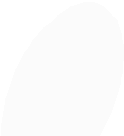
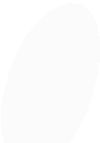
This is to certify that **Mr. Darshil Shah,** a 3rd year student at Manipal

Institute of Technology, Karnataka, has successfully completed his internship with Halara Labs Private Limited, reporting to the Head of Product team from **29th July 2020 to 6th November 2020.**

He completed his internship as a full stack developer, working on our mobile

& Web applications, under the project named Play Hatke, a real-money gaming mobile application.

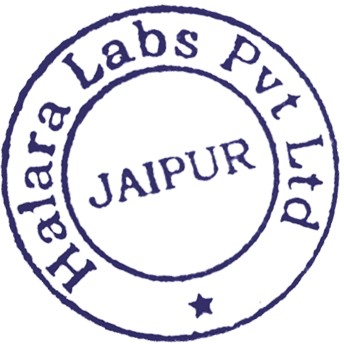
Mr. Darshil’s development skills are commendable. He has shown a willingness to learn & execute tasks and has been very sincere and dedicated to the work assigned to him.



*P:* 91+ 9413867113

*E:* [playgamesnow247@gmail.com](mailto:playgamesnow247@gmail.com)

*W:* [www.playhatke.com](http://www.playhatke.com/) *A*: F-02, Plot 12-13, Vatsalya residency, jagatpura, jaipur (raj), 302017.

Sincerely,

**Saket Bhardwaj**

Co-Founder

Halara Labs Pvt Ltd.

BHARDWAJ

# SAKET

Digitally signed by SAKET BHARDWAJ

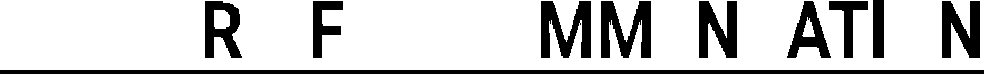
Date: 2020.12.11

20:49:26 +05'30'

HALARA LABS



CIN NO: U72900RJ2020PTC069333



Date: 12/12/2020

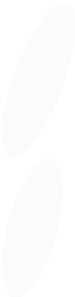
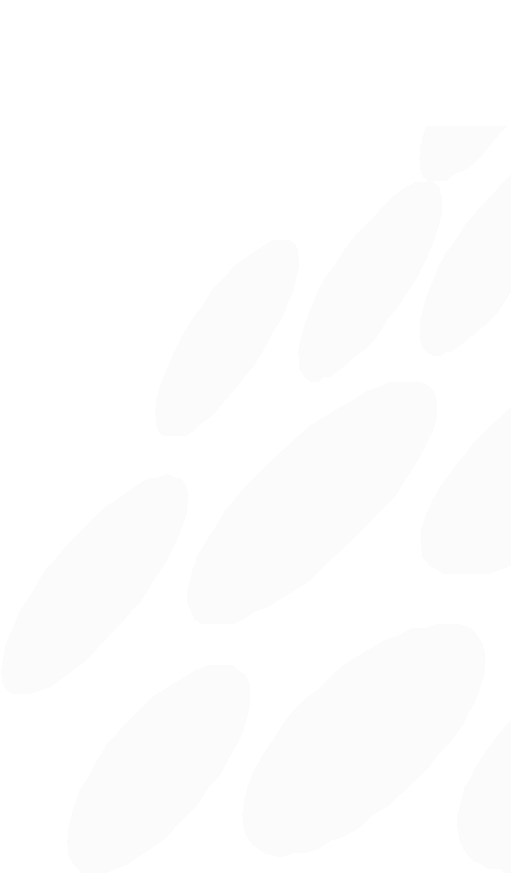
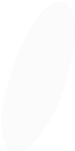
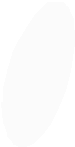
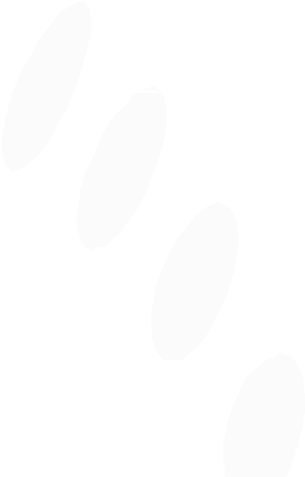
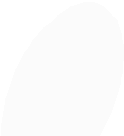
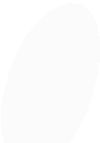
It gives me immense pleasure to recommend Mr. Darshil Shah, of B.Tech Computer Science Engineering 3rd year, Manipal Institute of Technology, Karnataka, India, for his future endeavors. We've known Mr. Darshil for

3 months from 29/7/2020 to 06/09/2020. During this time, Darshil worked as a full-stack app and web developer. We watched Darshil transform

into an excellent individual who excelled in both technological and interpersonal skills. We witnessed his strength to grasp and evaluate the new professional-level challenges, and then cleverly put them to real-world solutions. He created an application using Flutter and a web app using react for our startup which is a real-time gaming platform.

Darshil's achievements compete with his personal qualities and never giving up attitude. His compassion for his work is beyond his years. During this time, he coordinated very well with his fellow mate and made sure he gave the best of his skills. It was a great pleasure to work and grow with him.

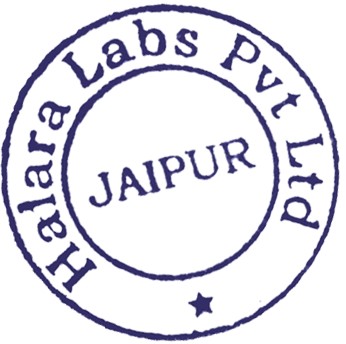
We wish him good luck for his future.



*P:* 91+ 9413867113

*E:* [playgamesnow247@gmail.com](mailto:playgamesnow247@gmail.com)

*W:* [www.playhatke.com](http://www.playhatke.com/) *A*: F-02, Plot 12-13, Vatsalya residency, jagatpura, jaipur (raj), 302017.



With Regards,

**Saket Bhardwaj**

Co-Founder

Halara Labs Pvt Ltd.

BHARDWAJ

# SAKET

Digitally signed by SAKET BHARDWAJ

Date: 2020.12.11

20:48:39 +05'30’

ACKNOWLEDGEMENT

I would like to thank my parents and my family members who have provided constant support throughout my life. Without them everything that I have achieved in life would have been impossible.

I would also like to thank my college for providing the necessary platforms so that I could get this internship. I would also like to thank my friends and my colleagues who guided me whenever I was stuck somewhere. I would also like to thank Intershala who provided the platform through which I got this internship.

Finally, I would like to thank Halara Labs Pvt. Ltd., for this wonderful opportunity of working for them. I would like to thank Shivam Agarwal, the director of Halara Labs for his continuous guidance and the faith he showed in me. I would like to thank Arnav Singh, my mentor for his constant support throughout the internship. Lastly, I am thankful to the entire Halara Labs team without whom this internship would not be possible.

I

ABSTRACT

I worked at Halara Labs as a Full Stack Web and App developer from August 2020 to November 2020. They make online games where users can play and earn real-money.

I built a fully-functional website from scratch which will be used by the team of Halara Labs to create and manage games. I used React.js along with basic HTML and CSS for the frontend and Node.js for the server which served as the backend. I used MySql and MongoDb as databases. This website is hosted on AWS. I also used Redis server for implementing the prize distribution after a game is completed which used bull queue to process the games in order of their completion time.

I also worked on building an android app which will be available on Play Store soon for users to download. I used Flutter and Dart as the frontend and build the basic functionality of the app. I used Node.js for the backend server along with MySql and MongoDb as the databases. I also worked with real-time transactions where users can withdraw or deposit their money using a third-party platform called razorpay.

II

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **Topic** | **Page No.** |
| 1 | About the organization | 1 |
| 2 | Background | 2 |
| 3 | Frontend – Website | 3 |
| 4 | Backend – Website | 5 |
| 5 | Databases | 7 |
| 6 | Frontend – App | 9 |
| 7 | Backend - App | 11 |
| 8 | Redis Server | 12 |
| 9 | Git | 13 |
| 10 | Conclusion | 14 |
| 11 | References | 15 |

III

About the Organization

Halara Labs Private Limited is a Rajasthan based Private company incorporated on 02 July 2020. It is classified as a Non-Government company and is registered at Registrar of Companies, Jaipur. It is involved in other computer related activities such as maintenance of websites of other firms/organizations or creation of multimedia presentations for other firms/organizations etc.

The company is a very early age startup. Playhatke is a flagship brand of Halara Labs. Currently, they are building real money skill – based gaming platform for the smartphone generation of the world. The app is completely safe and the company follows all the norms set by the Government of India involving exchange of real money. The basic idea of the company is to create mobile games for the users who invest their money and have a chance to win a huge sum depending upon the prize pool of the game, number of players, etc. Halara Labs tries to create games which are easier to play and can attract users at the same time. Playhatke app offers various skill based games to the player in various formats against other players in online leagues which are governed by different set of game rules. Players can join these leagues by paying an entre fee to win prizes. Currently there are four games namely Card Match, Unique Bid, Trivia and Chidiya Ud. It also creates some mega-games with a very huge prize pool and a 100% winning guarantee. Playhatke also provides a feature to the users to create their own games with a cap on the prize pool and compete with friends or family. Halara Labs continues to grow and prosper with new ideas developed and implemented every single day.

The app is still in production and is expected to be launched soon on the Play Store for android users.

(visit <https://www.playhatke.com/> for more details)

1

BACKGROUND

Before I was selected for this internship, I already had a pretty good knowledge of Web Development and could work with HTML, CSS and JavaScript. I also learnt Node.js and React.js, both of which are JavaScript frameworks extensively used in Web Development for developing web pages that are attractive, interactive and scalable. My entire internship (website part) was based on building a website using React.js and Node.js. I also worked with MySql which was taught as a course in my academic curriculum.

I even had a prior knowledge of App Development (android apps) and had made several projects using Dart and Flutter before starting this internship. Flutter is extensively used for android app development to make apps that are expressive, responsive and have a flexible UI. The basic Flutter code in based on Dart which is a programming language similar to JavaScript. My job during the last 1-2 months of the internship was developing the already existing android app and it was completely based on Flutter.

Although during the internship, this knowledge was not sufficient and I got to learn many new frameworks, technologies, etc. The most important and the most used was GitHub as I had to constantly push my code, so that my mentor could monitor my progress. I also learnt about Redis server which is an in-memory data structure store and a very fast server. I also learnt about MongoDb during the course of internship as it was used extensively along with MySql. I acquired knowledge about basics of Amazon Web Services (how to host a server on AWS) and how to release an android app on the Play Store.

2

FRONTEND – WEBSITE

The entire frontend of my website is completely made on React.js along with some HTML and CSS. React is a JavaScript library which is used for building user interfaces. React is Declarative i.e. it’s painless to create interactive UIs. React allows us to design simpler views for each state in the application, and then React will efficiently update and render only the components that have changed. React is component-based, hence I could design components that manage their own state and I could merge several components to make complex UIs. And as components are written in JavaScript instead of templates it becomes very easier to pass rich/complex data through the components. A component is divided into stateful and stateless components. I used stateless components for static pages which do not change and hence have to be rendered only once. In contrast I used Stateful components for those pages where data (state of the component) kept changing constantly. Hence whenever the state of the component changes, components which depend on the data of that state are needed to re-render and this is successfully implemented by the React DOM.

I used Material UI for most of the styling part in my website. Material UI has React components for faster and easier Web Development. It basically contains React components with styling or CSS applied to them and it can be directly integrated into any project without worrying much about the UI of the component. Using Material UI made my job relatively easier as I could style my components which were more attractive and I had to write very less code as compared to the traditional CSS code which would have to be used otherwise.

For example :

 function App () {

return (

<Button variant=”contained” color=”primary”>

Hello World

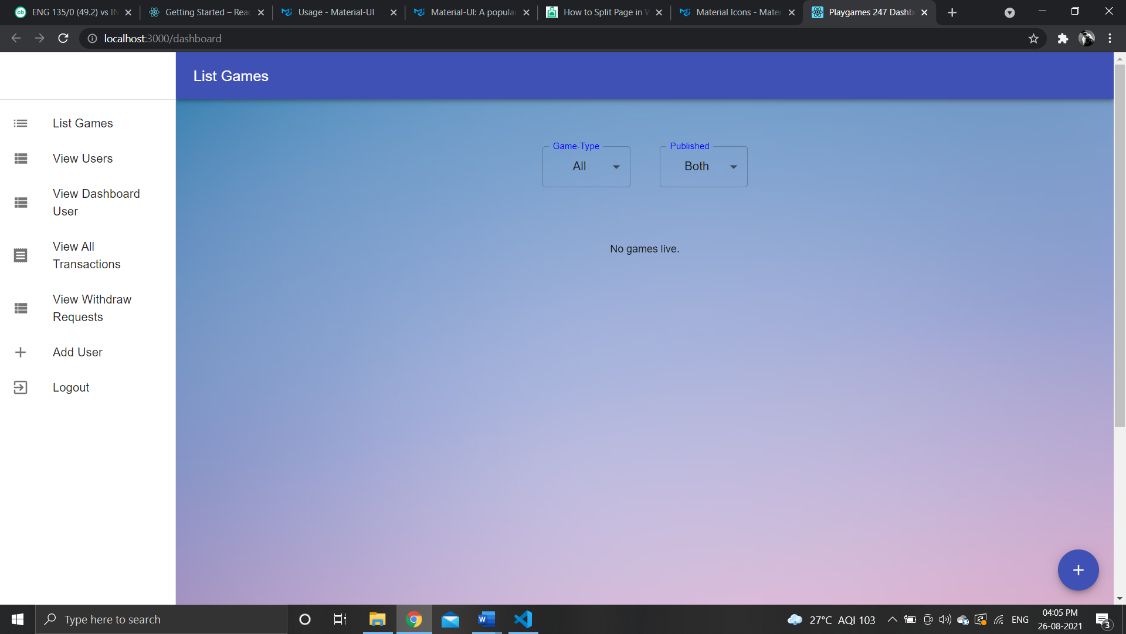
</Button>

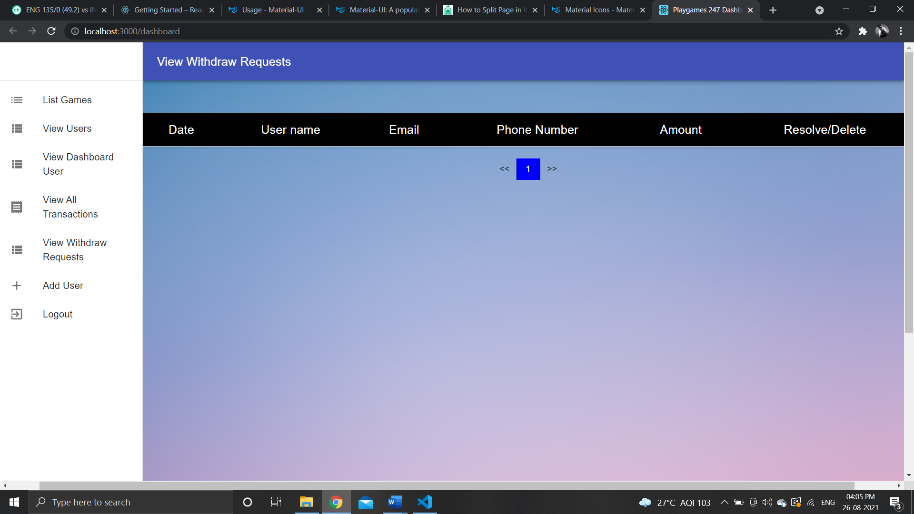
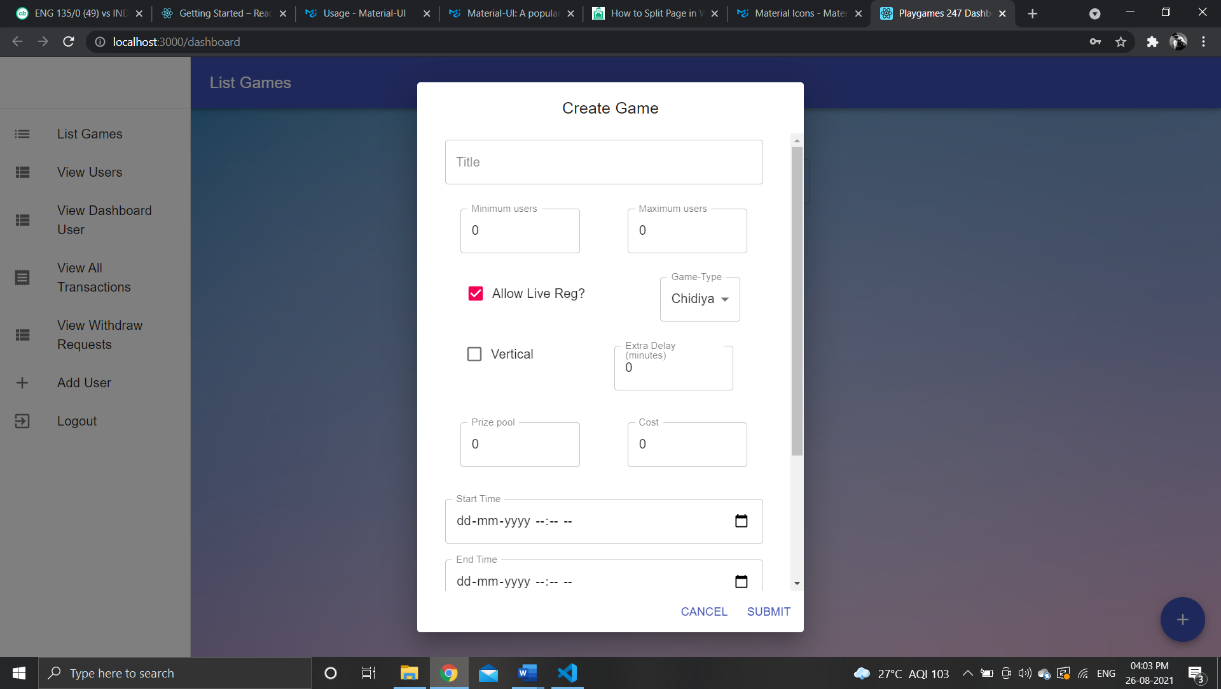
);

As the above code shows, just by adding two fields ‘variant’ and ‘color’ we can create a stylish button without literally writing any code.

I used many such components and integrated them in my website. Some of them are Appbar, Button, Divider, Drawer, List, ListItem, Modal, Table, TableBody, TableCell, Toolbar, etc. I also made use of Material UI Icons, which as the name suggests is used to add attractive icons for different webpages. These icons can be easily integrated into the React code which is JSX as they are written in the form of components. Hence it’s just like adding any another component in React.

3



These are some of the webpages from the website. (Can’t add images with actual data because of the policies of the company.)

I also used Axios which is a npm package (npm is explained in detail in the next section). Axios is a promise-based http client for the browser. I used it to implement the basic CRUD(Create, Read, Update, Delete) operations and make API calls to the backend server.

I also used the moment package from npm.I used this package to manipulate the DateTime object provided by JavaScript.

I used jwt-simple which is also a npm package. The company wanted an auto-login feature and to make this possible I encoded the basic user data during login with the help of this package and stored it in local storage of the browser which can be fetched every time user wants to sign into the dashboard.

4

BACKEND – WEBSITE

The complete backend of the website is expressed as Node.js and its packages. Node.js is a JavaScript framework extensively used as a backend server in Web Development. As an asynchronous event-driven JavaScript runtime, Node.js’s main advantage is to build scalable network applications. As a result of this it can handle many connections concurrently. For each individual connection, a callback is fired and a new socket is created which initiates the TCP connection between the client and the server. When there is no work Node.js goes to sleep reducing the load on the server.

Node Package Manager(npm) is a critical part of the JavaScript community. npm is the package manager for Node.js and has already developed packages whose functionality can be used while developing a Node.js server. npm is extremely configurable to support a variety of use cases. Most commonly, it is used to publish, discover, install and develop node programs.

Express is a npm package which is probably the most widely used npm package when a Node.js server is used. Its main philosophy is to provide small, robust tooling for HTTP servers. It can listen to multiple requests and respond to them with robustness from different clients at the same time.

const app = express()

This line of code creates a new express object which is later used for all express applications.

app.listen (Port, callback)

This line of code is the most important in entire Node.js application. It is used to bind and listen the connections on the specified host and port. After this the callback is fired wherein we can write any valid JavaScript code.

Finally there are app.get(), app.post(), app.update() and app.delete() functions to respond to client requests. As the name suggests, they are used to handle GET, POST, UPDATE and DELETE requests from the client. It allows using middlewares which as the name suggest are executed after the function is called and before the callback is called. I used a middleware every time an API call was made to verify if the user is authenticated or not. This helped in reducing the redundancy in the code as the middleware was called for almost every API call but the code for it was written only once. The callbacks in these methods have request and response parameters. The Request parameter is an object with body, params, query, etc. as the keys. The response parameter as the name suggests is used to send responses to the client. Some of the methods defined by response parameter are send(), redirect(), etc.

I used a bodyparser package provided by the npm. This package is used as a middleware to parse the data received from the client before it can be used by the callback function and is available under the req.body property. The most widely used format is bodyParser.json() which parses only JSON (Java Script Object Notation) data and JSON is the most convenient type to pass data between the frontend and the backend.

5

We used a package named cors which is again provided by the npm. CORS is short hand for Cross-Origin Resource Sharing. It is a mechanism to allow or restrict requested resources on a web server depending on where the HTTP request was initiated. This policy was used to secure our web server from access by another website or domain.

We had several API keys, database credentials, AWS credentials, etc. which were needed to be separated form the source code. Hence I stored this secret information in a file named .env to separate it from the source code. The idea behind this is that the source code might be shared with several developers, even uploaded on a server, so it was important to secure that information. I used a package named dotenv to excess these variables in the source code. The variables stored in the .env file can be accessed by the statement process.env.{variableName} .

Another package used for improving the security of the server is bcrypt. The bcrypt package makes it really easy to hash and compare passwords in Node.js. It hashes passwords using the hash() method, based on the number of salt rounds used. The hashed passwords are the stored in the database and while verifying the compare() method is used.

Another package similar to cors is helmet which is used to secure the express app (server) by setting various http headers. Although it’s not silver bullet, but it can help. We used the basic version of helmet – helmet(). This function is wrapper around 11 middlewares which are enabled by default.

I used a package named uuid to generate random unique ids which were stored as the primary key in the database. We used version 4 of the uuid package in which ids are created by Cryptographically-strong values and that’s why there is almost no chance that the same id will be generated again.

6

DATABASES

A total of 2 databases were used for the entire project. First one is MySQL which was used to store user data in general and the second one is MongoDb which was used to store any game data required for the project.

We used the MySQL database provided by the Amazon Web Service (AWS). Amazon RDS makes it easy to set up, operate and scale MySQL documents in the cloud. MySQL is an open-source relational database management system where data is stored in the form of tables consisting of rows and columns. Some of the features provided by the Amazon RDS are easy and managed deployments, fast and predictable storage, backup and recovery, monitoring and metrics, isolation and security, etc. We used a package named sequalize provide by the npm to create models of the tables. This was used as a template while inserting a row in a table, if the data is not in the form of the template it gives an error and does not store ambiguous data in the database. Some of the basic commands used are:

TableName.create(data) This query is used to enter data into the table.

TableName.findOne(query) This query is used to find a single row form the table which satisfies the query.

TableName.find(query) This query is used to find all rows form the table which satisfie the query.

TableName.deleteOne(query) This query is used to delete a single row from the table which satisfies the query.

TableName.count() This query is used to count the number of rows in the table.

MongoDb is a source-available cross-platform document-oriented database program. It is classified as a NoSQL databases program and uses JSON-like documents called BSON(binary JSON format ) with optional schemas. MongoDb can coordinate and control changes to the structure of the document using schema validation. Data stored in BSON can be searched and indexed, tremendously increasing performance. Some of the features provided by the MongoDb are flexible and intuitive data model, data as code, unlimited scalability, unified API, transparent data movement, etc. We used a package named mongoose provided by the npm to create schemas. Although MongoDb allows optional arguments, we used mongoose to maintain a uniformity throughout the database and because of this executing queries also becomes easier. Some of the basic commands used are:

CollectionName.create(data) This query is used to enter data into the collection.

Collection.find({},{},query) This query is used to find all the documents from the collection which satisfy the query.

Collection.count() This query is used to count the number of documents in the collection.

Collection.deleteOne(query) This query is used to delete a single document from the collection which satisfies the query.

Collection.updateOne(query,updateQuery) This query is used to update a single document to a new value(updateQuery) from the collection which satisfies the query.

7

Apart from the basic CRUD operations, I had to code some complex operations like pagination and transactions.

Pagination is a process in which data in the tables/collections can be split into individual pages, which is an efficient method of showing a large number of records in a small place. As it can be seen in one of the images form the frontend section, a large part of the frontend was divided into multiple pages. I had to implement pagination both in MySQL and MongoDb.

Pagination in MySQL:

Data = model.TableName.findAll({

offset: (pageNo-1)\*size,

limit: size

});

Pagination in MongoDb:

const query = {

limit: size,

skip: (pageNo-1)\*size

};

Collection.find({},{},query);

In both the above examples, size is the number of rows/documents in one page and pageNo corresponds to the current page which ranges between 1 and max\_page.

As my app involves real-time money, use of transactions was the most important thing in the entire project. Any logical calculation in a consistent mode in a database is called a transaction. Its job is to verify that the ACID (Atomicity, Consistency, Isolation, Durability) properties are always satisfied. I used transactions in both MySQL and MongoDb specially when I had to update anything related to the wallet (transaction).

Transactions in MySQL;

const t = db.sequelize.transaction();

t.rollback();

t.commit();

Transactions in MongoDb:

session = collection.startSession();

session.startTransaction();

session.abortTransaction();

session.commitTransaction();

session.endSession();

Whenever there is an error in any of the database application rollback() or abort() is called to go back to the starting point of the database and undo all the operations that have been performed on the database in the current session. When all the statements are successfully executed, commit() is called to make the changes in the database permanent.

8

FRONTEND – APP

The android app is completely built on dart and flutter (1.4) and uses Node.js as the backend for API calls. Dart is the programming language used to code Flutter apps. It is a language designed for client development, such as for the web and mobile apps. Dart is an object-oriented and platform-independent language. Flutter is an open-source UI software development kit created by Google for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Flutter, just like React has stateless and stateful widgets. Stateless widgets are static widgets which do not change dynamically. Stateful widgets are dynamic widgets and depend upon a state object, and these widgets recompile every time the state changes. Flutter has functions which can make the app automatically responsive which reduces the overhead to write extra Media Query code.

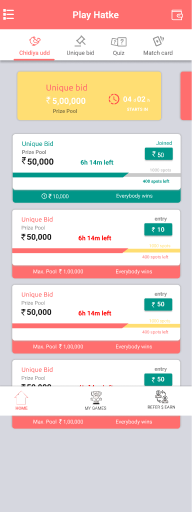
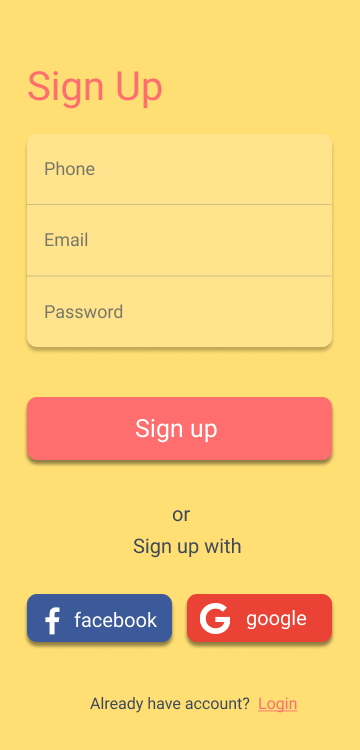
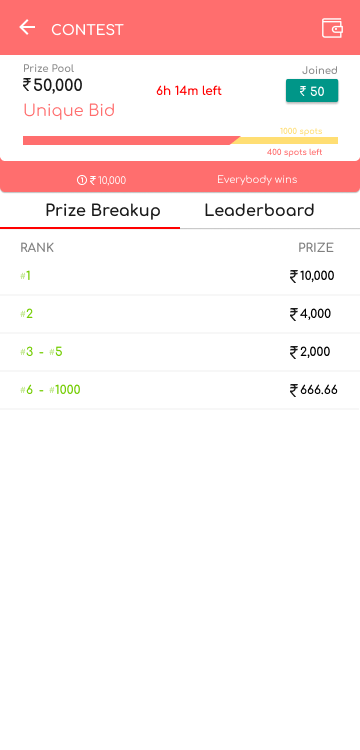
I used a large number of packages and widgets but one of the most important one was the Provider package. It is a wrapper around InheritedWidget (used to pass information down the widget tree) to make them easier to use and more reusable. The biggest advantage is that we can get any information anywhere in any widget and it is not necessary to pass it down the widget tree. We just need to wrap the root with the Provider of class which contains that information. This class is defined as follows :

class ClassName with ChangeNotifier{}

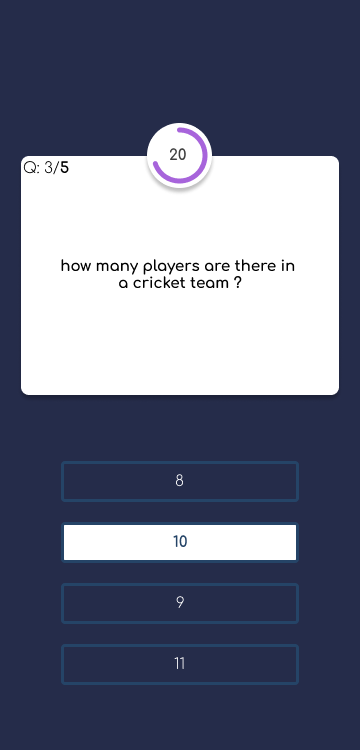
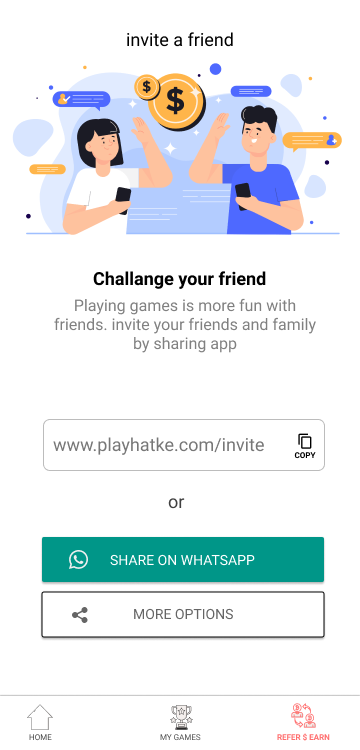
This also ensures that when the state of the class changes it is notified to all the widget who depend upon the data of that class, and flutter accordingly reloads all the necessary widgets.

To implement the autologin feature we used a class provided by Flutter called FlutterSecureStorage which stored the token as the key in the local storage of the user’s phone. Whenever user opens the app, this token was verified and if it was not set to Null (means the user is authenticated), the user was directly logged into the app. Similarly, during login, a token is created and is stored in the local storage for later use.

Some of the many screens designed are:

9

Some of the widgets used to implement these are: Scaffold, Container, Text, Column, Row, ListView, AlertDialog, BoxDecoration, ImageDecoration, TikTikTimer, GridView, Card, GestureDetector, TabBar, Expanded, RaisedButton, FlatButton, Padding, SingleChildScrollView, Drawer, etc and many more widgets. These widgets were very easy to implement and a detailed implementation of the same could be found on the official documentation of these widgets.

Apart from these widgets, I used a http package to make HTTP calls such as GET, POST, UPDATE and DELETE to the backend. I used flutter\_icons package to add icons such as share, wallet, whatsApp, copy, back, login, etc to my app. I used the font\_awesome\_flutter package to add stylish fonts to my app.

Finally I used the razorpay\_flutter package to make real-time payments possible. We used Razorpay, a third-party service to manage all the payments. On clicking the pay option, the user would be redirected to the razorpay portal when different payment options were available and on successful payment the user would be redirected back to the app with updated wallet.

10

BACKEND – APP

The backend of the app is also designed using Node.js where most of the functionalities and the technologies used are same as the backend of the website. Although some of the new technologies/packets that I used are NodeMailer, Razorpay API and redis server (explained in detail in the next section ). The databases used are also the same as the website with a very similar functionality.

There are functions in the app which require mail to be sent form the admin to the users, one of them is to resend otp on mail. So we used a package called NodeMailer provided by the npm. It is asynchronous in nature and runs on single threaded event loop. To send a mail using Nodemailer the basic steps to be followed are :

const transporter = nodemailer.createTransport({obj});

The obj variables contains details about the host, port, username and password.

Then declare a variable mailDetails that contains the sender and receiver Mail IDs, subject and the content of the mail.

I also used a npm package called email-templates which as the name suggests is used to create templates for the mails. It requires 2 files to be created and stored in the same folder, one of which is a subject file and the second one is the text file. While creating the mail object add the name of the folder as a string literal to the key ‘template’.

As mentioned in the previous section, we used Razorpay SDK to integrate with Razorpay Payment Gateway. Hence, we used a npm package razorpay in the backend to service requests from the app and direct them to the Razorpay SDK. The steps involved to initiate a transaction are:

const instance = new Razorpay({

key\_id:

key\_secret:

});

This creates a new instance of t Razorpay and the variables key\_id and key\_secret are available on the Razorpay website.

const options = {

amount:

currency:

receipt:

};

instance.orders.create(options,callback);

The options variable specifies the amount (in paise) in the transaction, currency refers to the type of currency used (INR in our case), and receipt generates a payment confirmation receipt. The next line calls the razorpay API with options as the object and initiates the transaction.

It also has a webhook function which needs to be implemented but it is pretty straightforward and can be done by referring to the official docs of the Razorpay SDK.

11

REDIS SERVER

Redis offers purpose-built in-memory data structures and operators to manage real-time data at high speed and scale. It can be used as database, cache and message broker. All the data is stored in RAM (because of which can’t store large amount of data), so the speed of the system is phenomenal. We used Redis Server for processing the prizes once the game is completed and since a large number of games might be live at the same time, use of conventional databases was not feasible. We used the Redis Server provided by the AWS. I had no prior knowledge of Redis Server and had learn about it from scratch.

taskCollection = db.collection(collectionName);

changeStream = taskCollection.watch([],{});

changeStream.on(‘change’,callback);

These lines of code add a listener to a collection and an event is fired every time a document is updated or a new document is created.

We used a npm package called Bull which implements a fast and robust queue system based on redis. We used a priority-based queue with the game finish time as the priority.

Queue = require(‘bull’);

obj = new Queue(name, redisServerUrl);

obj.process(function(job,done));

obj is a new queue object which runs on the redis server. The next line is called by the callback in the changeStream,on function to add the current job in the queue. A function is call is made to calculate the prizes and form the leaderboard when the job is popped from the queue.

options = {

dif : gameEndTime - currentTime

};

job = obj.add({data, options});

The options variable defines the priority of the current job and then it is pushed into the queue based on that priority. The obj.add() method is used to add the current job into the queue.

12

GIT

We made extensive use of GitHub in our project. It is a provider of internet hosting for software development and version control using Git. Git is a software for tracking changes in any set of files, usually among programmers collaboratively developing code during software development. I had some basic knowledge of Git but I learnt a lot more about it during the course of my internship. The version control provided by Git is extremely important while working in teams. Apart form these it allows provides a feature to backtrack to a previous version and access it, in case there are some issues in the current version. Not only these, but it allows developers in future who will be working on the same project to see how it was built from scratch for a better understanding of the project.

Some of the git commands I used during the course of the internship are:

git init – This initializes an empty repository in the current working directory.

git add . – This adds the newly created/modified files to the staging area.

git commit – This takes the staged snapshot and commits it to the project history.

git push – This moves the local branch to another repository. It sends all the commits in the project history which are not pushed to the repository.

git fetch – This fetches a branch from another repository to the current repository. It allows users to inspect the changes before merging it with the project.

git log – This displays the logs(previous commits and the changes) of the project.

git rebase -I – This is used when the master repository is ahead than the current repository and merges the code with the current repository so that both the repositories become in-line with each other.

git branch – This creates a new branch in the same repository.

git checkout – This is used to navigate to different branches in the same repository.

13

CONCLUSION

It was an amazing experience to work with Halara Labs Private Limited under the mentorship of Arnav Singh. I could gain further knowledge in the fields of Web Development, App Development and GitHub. This was my first internship and I was pretty excited to begin and even after finishing my internship, my excitement remained the same. I gained further insights on how the corporate industry works and also how to be productive while working from home. I learnt that finishing tasks before the deadlines is very much important irrespective of the number of hours worked. I learnt how to resolves issues, whenever I got stuck with something, it’s important to stay calm and think of a solution instead of worrying and panicking about it. It’s important to make the best use of resources provided to you. Although the app didn’t release while I was working at the organization as some minor changes were left and are being done by other developers, the experience that I got in this organization is above all and I am extremely grateful to Shivam Agarwal for giving me this opportunity. I hope to work again with Halara Labs in the near future, if possible.

14

REFERENCES

1) https://www.mongodb.com

2) https://reactjs.org/docs/getting-started.html

3) https://flutter.dev/docs

4) https://nodejs.org/en/docs

15