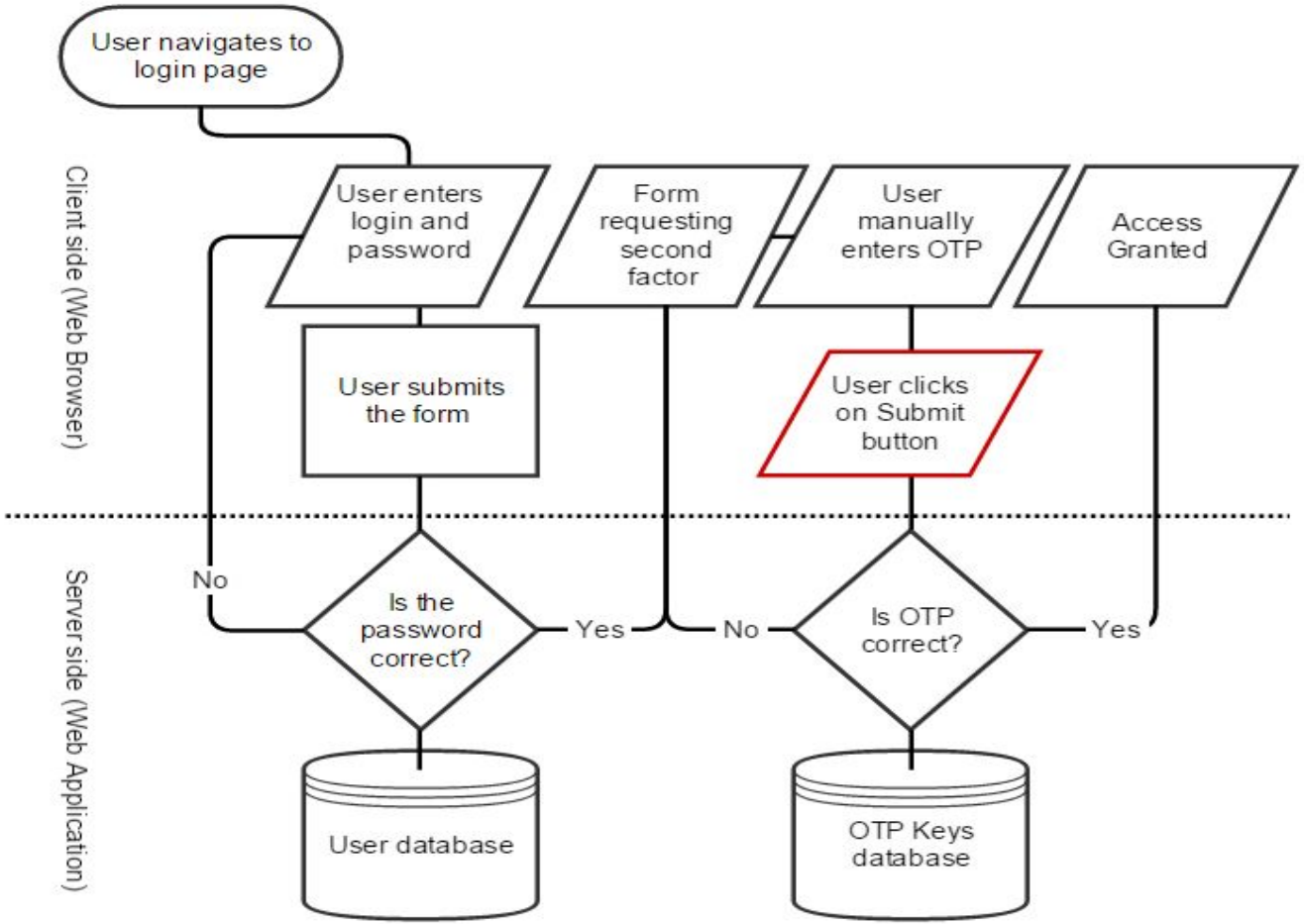
2 factor authentication

IT017- Kiran Chaudhary
IT011- Ishita Bhalodia

Tech stack

Back-end tech stack	nodejs(v16), express(v4)
Front-end tech stack	Html, css, javascript
Messaging queue	rabbitmq
For mail	nodemailer

Client and server workflow



File Structures

```
├── app.js
├── .env
└── src
    ├── config
    │   └── db.js
    ├── constant
    │   └── constant.js
    ├── database
    │   └── user.js
    ...
    ├── modal
    │   ├── verifymodal.js
    │   └── authmodal.js
    ├── schema
    │   ├── verifyschema.js
    │   └── authschema.js
    └── utility
        ├── consumer.js
        └── emailservice.js
```

Main route and app.js

```
// configuations
env.config();
const app = express();
const port = 3000;
// rabbitmq's consumer running
import "./src/utility/consumer.js";
// database connection
db();
app.listen(port, () => {
  console.log("port working");
});
// middleware
app.use(express.json({ limit: "100mb" }));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

```
app.use("/static", express.static(
  path.join(__dirname, "public")));
app.use(cookieParser());
app.use(cors());

app.post("/", function (req, res) {
  const { IsLogIn } = req.body;
  jwt.verify(IsLogIn, process.env.JWT_SECRET,
             (err, verifiedJwt) => {
    if (err) {
      res.status(400).send("no user login");
    } else {
      res.status(200).send("success");}
  });
});
app.use("/login", authrouter);
app.use("/login/verify", verifyrouter);
```

Auth route

```
const authrouter = express.Router();
authrouter.route("/").post(authPost);


async function authPost(req, res) {
  const { username, password } = req.body;
  const otp = otpGenerator.generate(6, {
    lowerCaseAlphabets: false,
    upperCaseAlphabets: false,
    specialChars: false,
  });
  const user = await
 findUserByUsername(username);
  if (!user) {
    res.status(400).json({
      result: "failed to  found user",
      message: "user not found on database",
    });
  } else {
    const match = await bcrypt.compare(password,
user.password);
    if (match) {
      res.app.set("email", { email: user.email
});
```

```
      const User = await
findUserByEmailForOto(user.email);

      if (!User) {
        saveNewOtpForUser(user.email, otp);
      } else {
        console.log("user existed");
        findUserAndUpdate(user.email, otp);
      }
      res.status(200).send("sucess");
    } else {
      res.status(400).send("password not
matched");
    }
  }
}


export default authrouter;
```

Verify route

```
const verifyrouter = express.Router();
verifyrouter.route("/").post(verifyPost);

async function verifyPost(req, res) {
  const { otp } = req.body;
  const { email } = res.app.get("email");

  const user = await
findUserByEmailForOto(email);

  if (!user) {
    console.log("user not found");
    res.status(404).json({
      result: "failed to  found user",
      message: "user not found on database",
    });
  }
  else if (user.expired <= Date.now()) {
    await findEmailAndDelete(email);
    res.status(403).json({
      result: "failed",
      message: "otp expired",
    });
  }
```

```
  else {
    const match = await bcrypt.compare(otp,
user.otp);
    await findEmailAndDelete(email);
    if (match) {
      const token = jwt.sign({ email: user },
process.env.JWT_SECRET, {
        expiresIn: "24h",
      });
      res.status(200).send({
        ok: true,
        token: token,
        message: "USER_LOGIN_SUCCESS",
      });
    } else {
      res.status(400).json({
        result: "failed",
        message: "not same otp",
      });
    }
  }
}
```

Schema and modals

```
const loginSchema = new Schema({
  username: {
    type: String,
    required: true,
    unique: true,
  },
  email: {
    type: String,
    minLength: 10,
    required: true,
    lowercase: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  created: {
    type: String,
    default: new Date().toISOString(),
  },
  lastActive: {
    type: String,
    required: false,
  },
});

const UserModal = mongoose.model("User",
loginSchema);
export default UserModal;
```

```
const VerifySchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true,
  },
  otp: {
    type: String,
    required: true,
  },
  created: {
    type: String,
    required: true,
  },
  expired: {
    type: String,
    required: true,
  },
});

const VerifyModal = mongoose.model("Verify",
VerifySchema);
export default VerifyModal;
```

Database functions

```
export const findUserByUsername = async
(username) => {
  const User = await UserModal.findOne({
    username,
  });
  return User;
};

export const findUserByEmailForOto = async
(email) => {
  const User = await VerifyModal.findOne({
    email,
  });
  return User;
};

export const findEmailAndDelete = async (email)
=> {
  await VerifyModal.findOneAndDelete({
    email,
  })
    .then(() => {
      return "success";
    })
    .catch((e) => {
      return "failed" + e;
    });
};
```

```
export const saveNewOtpForUser = async (email,
otp) => {
  const newOTP = new VerifyModal({
    email: email,
    otp: bcrypt.hashSync(otp, 10),
    created: Date.now(),
    expired: Date.now() + 120000,
  });

  await newOTP
    .save()
    .then(() => {
     addToQueue(email,otp);d
      return "success";
    })
    .catch((e) => {
      return "failed" + e;
    });
};

export const findUserAndUpdate = async (email,
otp) => {
  await VerifyModal.findOneAndUpdate(
    {
      email,
    },
    { expired: Date.now() + 60000, otp:
bcrypt.hashSync(otp, 10) }
  )
    .then(() => {
      addToQueue(email,otp);
      return "sucess";
    })
    .catch((e) => {
      return "failed" + e;
    });
};
```

Database

```
import mongoose from "mongoose";
import env from "dotenv";
env.config();
const options = {
  connectTimeoutMS: 5000,
  useNewUrlParser: true,
  useUnifiedTopology: true,
};
mongoose.set("strictQuery", false);
const db = async () => {
  mongoose
    .connect(process.env.MONGO_URI, options)
    .then(() => {
      console.log("mangodb connected");
    })
    .catch(() => {
      console.log("error while connecting");
    });
};

export default db;
```

Producer

```
const addToQueue=async(email,otp)=>{
  amqp.connect(process.env.AMQP_URI, function
(error0, connection) {
        if (error0) {
          throw error0;
        }
        connection.createChannel(function
(error1, channel) {
          if (error1) {
            throw error1;
          }

          channel.assertQueue("LogIn_Queue", {
            durable: false,
          });

          channel.sendToQueue(
            "LogIn_Queue",
            Buffer.from(JSON.stringify({ to:
email, OTP: otp }))
          );
        });
      });
}
```

consumer

```
import amqp from "amqplib/callback_api.js";
import env from "dotenv";
import sendMail from "./emailServices.js";
env.config();

amqp.connect(process.env.AMQP_URI, function
(error0, connection) {
  if (error0) {
    throw error0;
  }

  connection.createChannel(function (error1,
channel) {
    if (error1) {
      throw error1;
    }

    channel.assertQueue("LogIn_Queue", {
      durable: false,
    });
    console.log("consumer started");
    channel.consume(
      "LogIn_Queue",
      function (msg) {
        let Msg =
JSON.parse(msg.content.toString());
        sendMail(Msg);
      },
      {
        noAck: true,
      }
    );
  });
});
```

nodemailer

```
import nodemailer from "nodemailer";
import env from "dotenv";
env.config();
const transport = nodemailer.createTransport({
  service: "gmail",
  auth: {
    user: process.env.GMAIL_EMAIL,
    pass: process.env.GMAIL_PASSWORD,
  },
});
```

```
const sendMail = async (params) => {
  console.log(process.env.EMAIL);
  transport.sendMail(
    {
      from: process.env.GMAIL_EMAIL,
      to: params.to,
      subject: "verification otp",
      html: `
        <div
          class="container"
          style="max-width: 90%; margin: auto;
padding-top: 20px"
        >
          <h2>Thanks for using 2 Factor
authentication</h2>
          <p style="margin-bottom: 30px;">Pleas
enter the sign up OTP to get started</p>
          <h1 style="font-size: 40px; letter-
spacing: 2px; text-align:center;">${params.OTP}
</h1>
      </div>
        `,
    },
    (err, response) => {
      if (err) {
        console.log("error ", err);
      } else {
        console.log("responss", response);
      }
    }
  );
};
export default sendMail;
```

Client (index.html)

```
<body onload="redirecttonewlocation()">
  <div class="login-page">
    <div class="form">
      <h1 id="header">
        You are suceess fully passed 2 factor authentication
      </h1>
    </div>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/js-cookie/3.0.1/js.cookie.
  min.js"></script>
  <script>
    async function redirecttonewlocation() {
      // const IsLogIn = Cookies.get("IsLogIn");
      await axios
        .post("http://localhost:3000/", { IsLogIn: Cookies.get("IsLogIn") })
        .then((response) => {
          console.log(response);
        })
        .catch((error) => {
          window.location.href = "login.html";
        });
    }
    // redirect();
  </script>
</body>
```

Client (login.html)

```
<script>
  const btn = document.getElementById("button");
  function getfocus() {
    document.getElementById("error").innerHTML = "";
  }
  btn.addEventListener("click", async function (e) {
    e.preventDefault();
    const username = document.querySelector("input[type='text']").value;
    const password = document.querySelector("input[type='password']").value;
    if (username.trim() == "" || password.trim() == "") {
      document.getElementById("error").innerHTML = "fill form";
    } else {
      await axios
        .post("http://localhost:3000/login", { username, password })
        .then((response) => {
          console.log(response);
          window.location.href = "verify.html";
        })
        .catch((error) => {
          console.log(error.response.status);
          if (error.response.status == 400) {
            document.getElementById("error").innerHTML =
              "you entered wrong password";
          } else if (error.response.status == 404) {
            document.getElementById("error").innerHTML =
              "there is not username ";
          } else if (error.response.status == 300) {
          } else {
            document.getElementById("error").innerHTML =
              "there may server or other type of error";
          }
        });
    }
  });
```

Client (verify.html)

```
<script>
    document.getElementById("error").innerHTML = "";
  }
  btn.addEventListener("click", async function (e) {
    e.preventDefault();
    const otp = document.querySelector("input[type='text']").value;
    console.log(otp);
    await axios
      .post("http://localhost:3000/login/verify", { otp })
      .then((response) => {
        console.log(response.data.token);
        // console.log(response);
        const d = new Date();
        d.setTime();
        let expires = "expires=" + d.toUTCString();
        Cookies.set("IsLogIn", response.data.token, {
          expires: d.getTime() + 24 * 60 * 60 * 1000,
        });
        window.location.href = "index.html";
        // document.cookie = "islgin=jfa";
      })
      .catch((error) => {
        if (error.response.status == 400) {
          document.getElementById("error").innerHTML =
            "you entered wrong otp";
        } else if (error.response.status == 403) {
          document.getElementById("error").innerHTML = "otp expired ";
        } else {
          document.getElementById("error").innerHTML =
            "there may server or other type of error";
        }
      });
  });
</script>
```