BEAUTY BOOKING

Mariam CHARRADA #C21

OUTLINE

01

Context

The story behind
BEAUTY BOOKING

TEAM

- ❏ Project Manager
- ❏ Developer
- ❏ Designer
- ❏ Tester
- ❏ Popcorn Lover

CHALLENGES

- Understanding new frameworks
- Time management
- Endless errors!

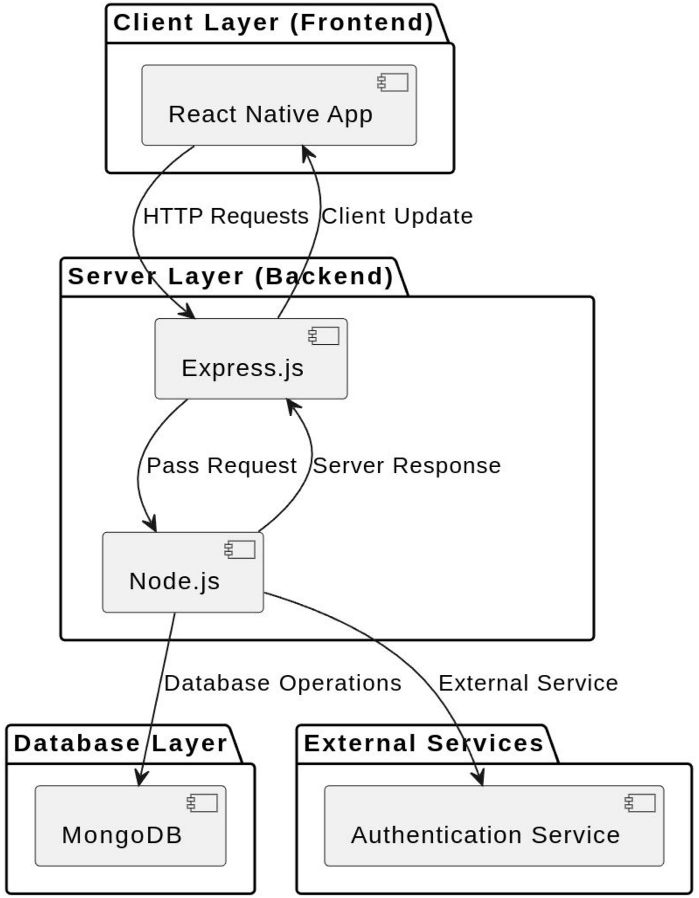
TECHNOLOGIES

Mongo DB
Database

Node js
Running Environment (backend)

Express js
Backend framework

React Native(Expo)
Frontend framework



ARCHITECTURE

- ➔ Client Layer
- ➔ Server Layer
- ➔ Database Layer

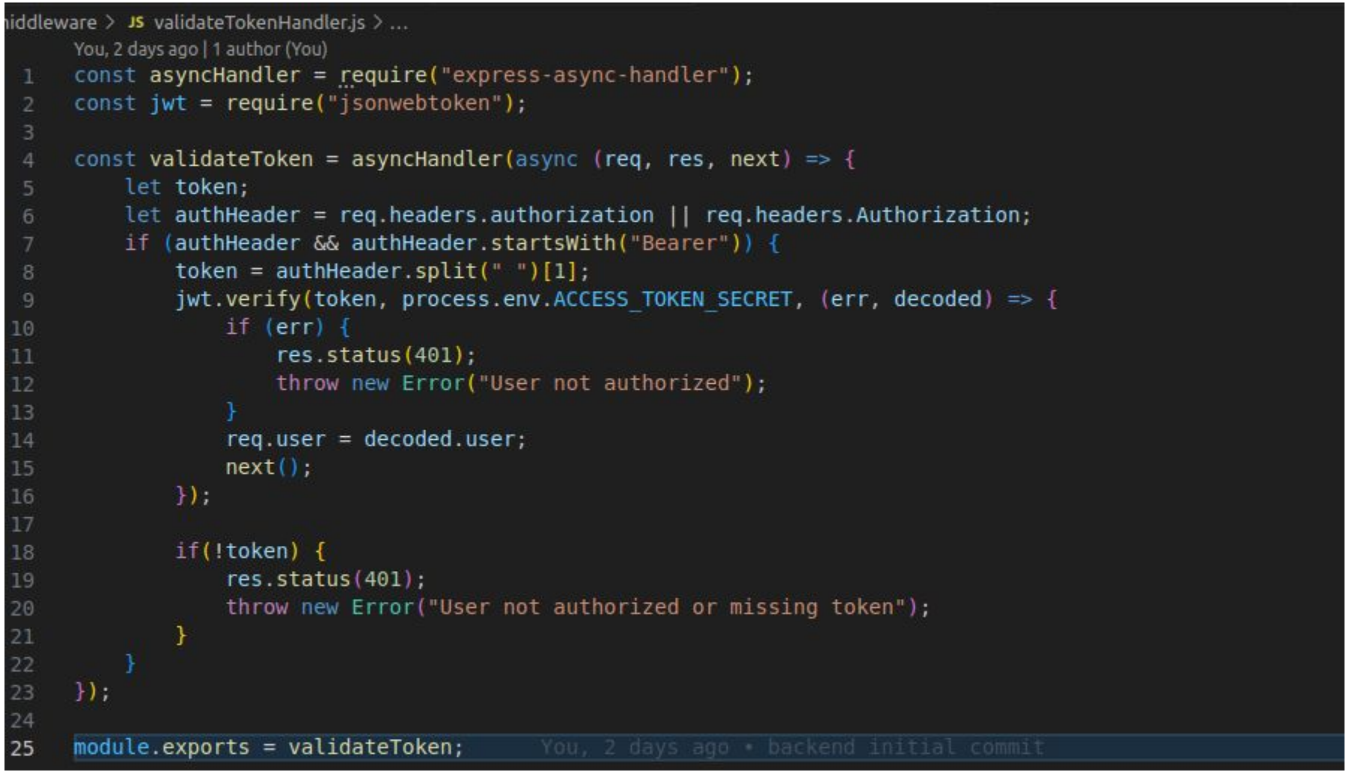
CORE ALGORITHMS

Authentication & Password Hashing

Registration

Appointment Booking

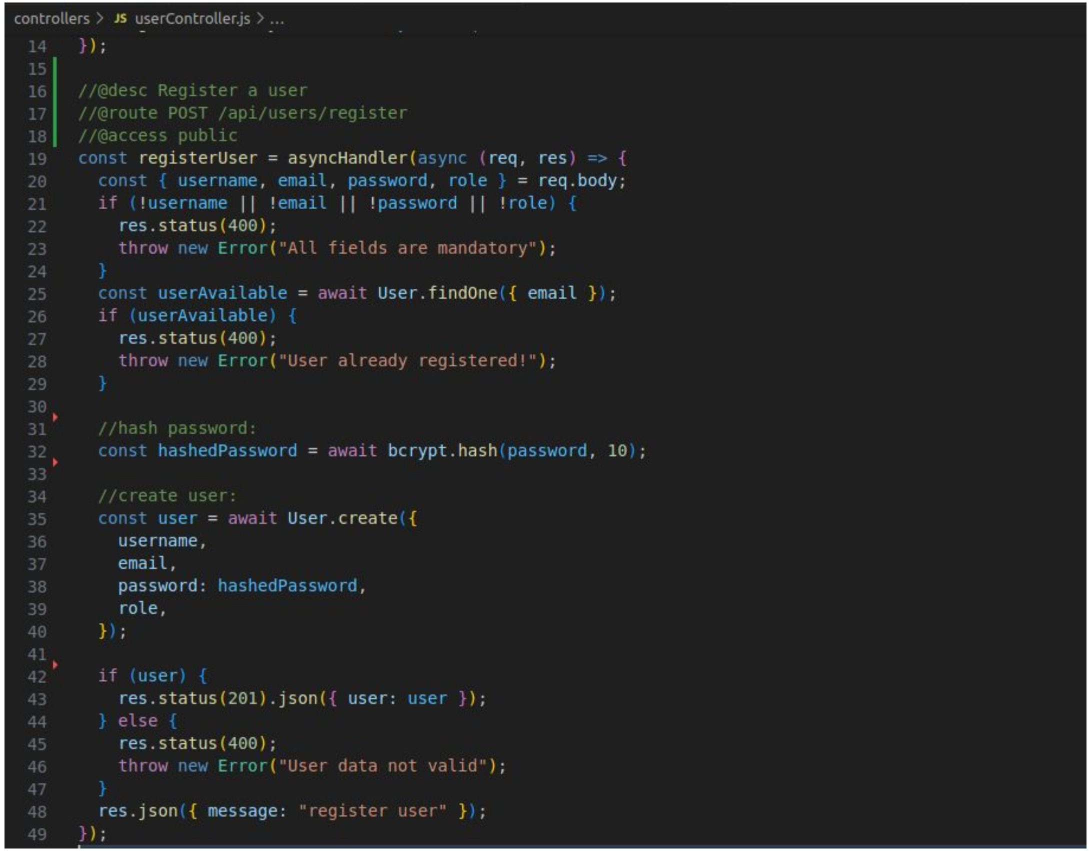
Authentication & Password Hashing


iddleware > JS validateTokenHandler.js > ...

You, 2 days ago | 1 author (You)

```javascript
const asyncHandler = require("express-async-handler");
const jwt = require("jsonwebtoken");

const validateToken = asyncHandler(async (req, res, next) => {
    let token;
    let authHeader = req.headers.authorization || req.headers.Authorization;
    if (authHeader && authHeader.startsWith("Bearer")) {
        token = authHeader.split(" ")[1];
        jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, decoded) => {
            if (err) {
                res.status(401);
                throw new Error("User not authorized");
            }
            req.user = decoded.user;
            next();
        });

        if(!token) {
            res.status(401);
            throw new Error("User not authorized or missing token");
        }
    }
});

module.exports = validateToken;      You, 2 days ago • backend initial commit
```

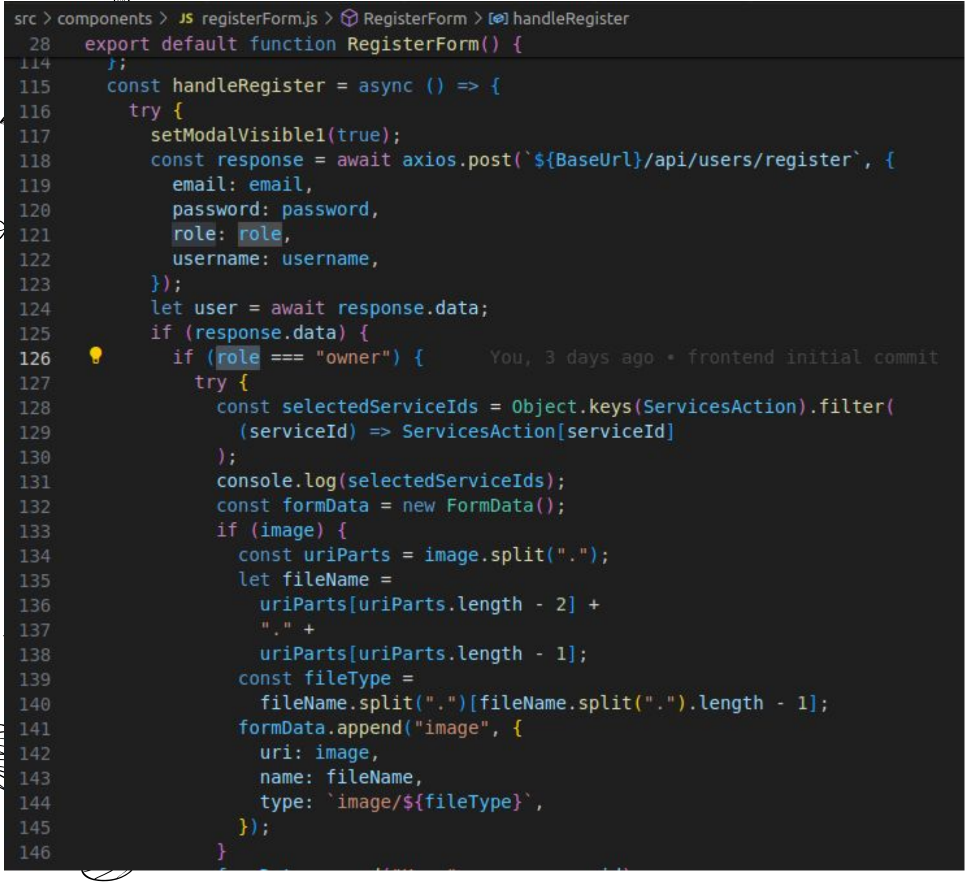
Authentication & Password Hashing



controllers > JS userController.js > ...

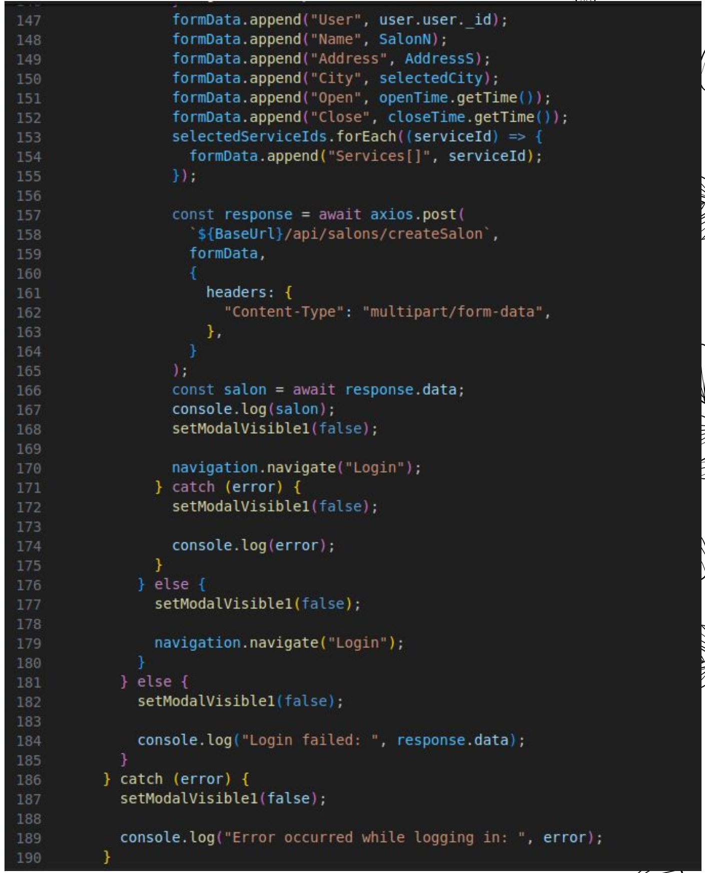
```
});

//@desc Register a user
//@route POST /api/users/register
//@access public
const registerUser = asyncHandler(async (req, res) => {
  const { username, email, password, role } = req.body;
  if (!username || !email || !password || !role) {
    res.status(400);
    throw new Error("All fields are mandatory");
  }
  const userAvailable = await User.findOne({ email });
  if (userAvailable) {
    res.status(400);
    throw new Error("User already registered!");
  }

  //hash password:
  const hashedPassword = await bcrypt.hash(password, 10);

  //create user:
  const user = await User.create({
    username,
    email,
    password: hashedPassword,
    role,
  });

  if (user) {
    res.status(201).json({ user: user });
  } else {
    res.status(400);
    throw new Error("User data not valid");
  }
  res.json({ message: "register user" });
});
```

Double Registration



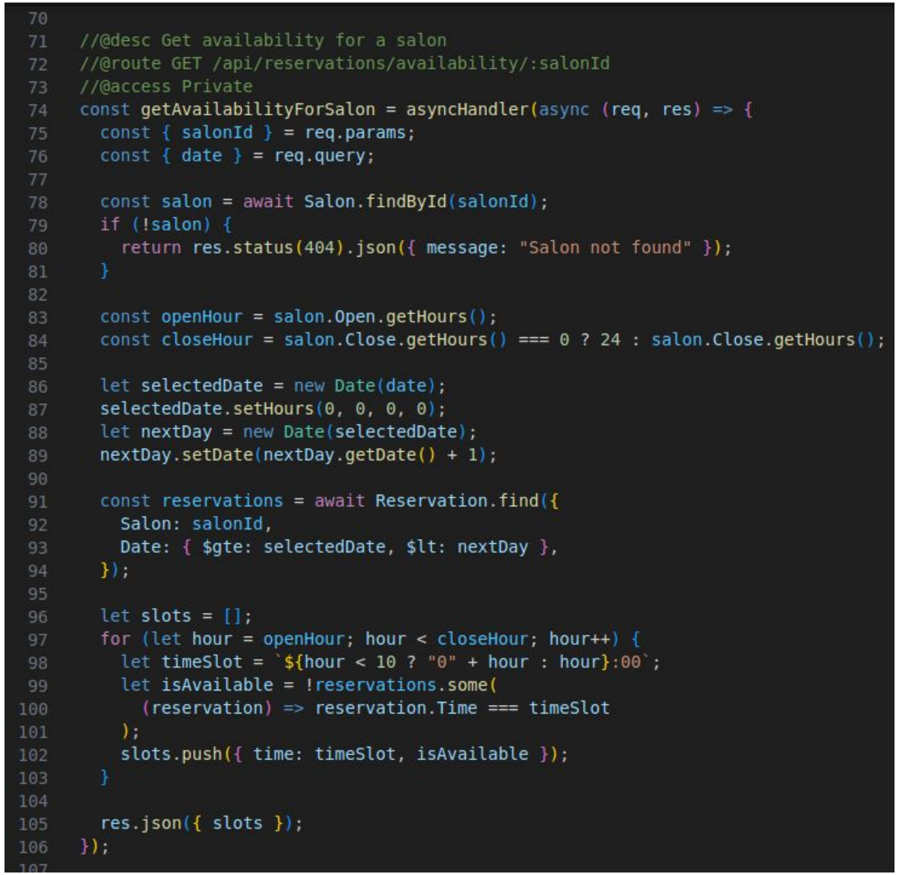
src > components > JS registerForm.js > RegisterForm > handleRegister

```javascript
export default function RegisterForm() {
  };
  const handleRegister = async () => {
    try {
      setModalVisible1(true);
      const response = await axios.post(`${BaseUrl}/api/users/register`, {
        email: email,
        password: password,
        role: role,
        username: username,
      });
      let user = await response.data;
      if (response.data) {
        if (role === "owner") {      You, 3 days ago • frontend initial commit
          try {
            const selectedServiceIds = Object.keys(ServicesAction).filter(
              (serviceId) => ServicesAction[serviceId]
            );
            console.log(selectedServiceIds);
            const formData = new FormData();
            if (image) {
              const uriParts = image.split(".");
              let fileName =
                uriParts[uriParts.length - 2] +
                "." +
                uriParts[uriParts.length - 1];
              const fileType =
                fileName.split(".")[fileName.split(".").length - 1];
              formData.append("image", {
                uri: image,
                name: fileName,
                type: `image/${fileType}`,
              });
            }
```



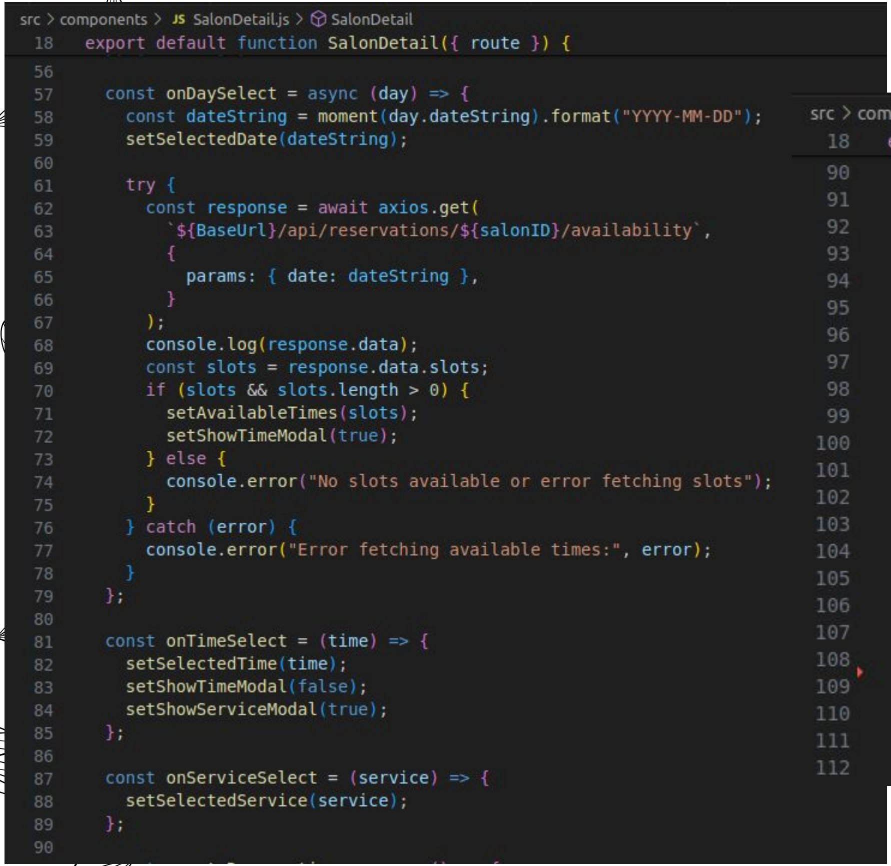
```javascript
            formData.append("User", user.user._id);
            formData.append("Name", SalonN);
            formData.append("Address", AddressS);
            formData.append("City", selectedCity);
            formData.append("Open", openTime.getTime());
            formData.append("Close", closeTime.getTime());
            selectedServiceIds.forEach((serviceId) => {
              formData.append("Services[]", serviceId);
            });

            const response = await axios.post(
              `${BaseUrl}/api/salons/createSalon`,
              formData,
              {
                headers: {
                  "Content-Type": "multipart/form-data",
                },
              }
            );
            const salon = await response.data;
            console.log(salon);
            setModalVisible1(false);

            navigation.navigate("Login");
          } catch (error) {
            setModalVisible1(false);

            console.log(error);
          }
        } else {
          setModalVisible1(false);

          navigation.navigate("Login");
        }
      } else {
        setModalVisible1(false);

        console.log("Login failed: ", response.data);
      }
    } catch (error) {
      setModalVisible1(false);

      console.log("Error occurred while logging in: ", error);
    }
```

Appointment Booking (Node js)



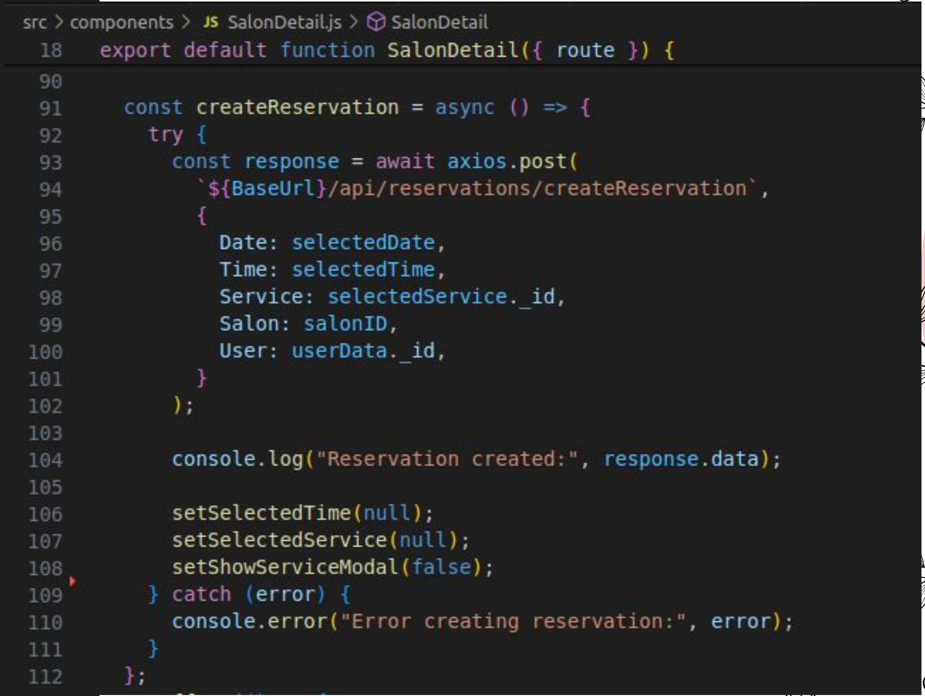
```
//@desc Get availability for a salon
//@route GET /api/reservations/availability/:salonId
//@access Private
const getAvailabilityForSalon = asyncHandler(async (req, res) => {
  const { salonId } = req.params;
  const { date } = req.query;

  const salon = await Salon.findById(salonId);
  if (!salon) {
    return res.status(404).json({ message: "Salon not found" });
  }

  const openHour = salon.Open.getHours();
  const closeHour = salon.Close.getHours() === 0 ? 24 : salon.Close.getHours();

  let selectedDate = new Date(date);
  selectedDate.setHours(0, 0, 0, 0);
  let nextDay = new Date(selectedDate);
  nextDay.setDate(nextDay.getDate() + 1);

  const reservations = await Reservation.find({
    Salon: salonId,
    Date: { $gte: selectedDate, $lt: nextDay },
  });

  let slots = [];
  for (let hour = openHour; hour < closeHour; hour++) {
    let timeSlot = `${hour < 10 ? "0" + hour : hour}:00`;
    let isAvailable = !reservations.some(
      (reservation) => reservation.Time === timeSlot
    );
    slots.push({ time: timeSlot, isAvailable });
  }

  res.json({ slots });
});
```

Appointment Booking (RN)



src > components > JS SalonDetail.js > SalonDetail

```javascript
export default function SalonDetail({ route }) {

  const onDaySelect = async (day) => {
    const dateString = moment(day.dateString).format("YYYY-MM-DD");
    setSelectedDate(dateString);

    try {
      const response = await axios.get(
        `${BaseUrl}/api/reservations/${salonID}/availability`,
        {
          params: { date: dateString },
        }
      );
      console.log(response.data);
      const slots = response.data.slots;
      if (slots && slots.length > 0) {
        setAvailableTimes(slots);
        setShowTimeModal(true);
      } else {
        console.error("No slots available or error fetching slots");
      }
    } catch (error) {
      console.error("Error fetching available times:", error);
    }
  };

  const onTimeSelect = (time) => {
    setSelectedTime(time);
    setShowTimeModal(false);
    setShowServiceModal(true);
  };

  const onServiceSelect = (service) => {
    setSelectedService(service);
  };

```



src > components > JS SalonDetail.js > SalonDetail

```javascript
export default function SalonDetail({ route }) {

  const createReservation = async () => {
    try {
      const response = await axios.post(
        `${BaseUrl}/api/reservations/createReservation`,
        {
          Date: selectedDate,
          Time: selectedTime,
          Service: selectedService._id,
          Salon: salonID,
          User: userData._id,
        }
      );

      console.log("Reservation created:", response.data);

      setSelectedTime(null);
      setSelectedService(null);
      setShowServiceModal(false);
    } catch (error) {
      console.error("Error creating reservation:", error);
    }
  };
```

PROCESS & TIMELINE

WEEK 1+2

Learning

WEEK 3+4

Express
+
Api testing
+
RN Screens

WEEK 4-6

Api
integration
+
Test &
validation

THANKS!

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

info@slidesgo.com
+91 620 421 838
yourcompany.com

Please keep this slide for attribution