Department of Computing

Bachelor of Science (Hons) in Software Development

Creche Connect

Technical Manual

Supervisor: Chris Meudec
Student Name: Michal Gornicki
Student Number: C00265618
Date: 2022/2023

Overview

The purpose of this manual is to display my Creche Connect code. The structured code for this application will display all of the components needed to build this app.

My code is available on GitHub: https://github.com/m1ner79/project-cc

Table of Contents

1. AddChild.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState} from "react";
import {Container, Form, Button, Card} from "react-bootstrap";
import {collection, addDoc} from "firebase/firestore";
import {db} from "../firebase";
import Navigation from "./Navigation";
import {Link} from "react-router-dom";
import {doc, updateDoc} from "firebase/firestore";

const AddChild = () => {
    const [firstName, setFirstName] = useState("");
    const [lastName, setLastName] = useState("");
    const [parentName, setParentName] = useState("");
    const [dob, setDob] = useState("");
    const [parentEmail, setParentEmail] = useState("");
    const [parentMobile, setParentMobile] = useState("");
    const [healthInfo, setHealthInfo] = useState("");
    const [additionalInfo, setAdditionalInfo] = useState("");
    const [childId, setChildId] = useState("");
    const lowFirstName = firstName.toLowerCase();
    const lowLastName = lastName.toLowerCase();
    const [updateMode, setUpdateMode] = useState(false);

    const buildSearchArray = (searchTerm) => {
        const searchTerms = [];
```

```
    let counter = 0;
    let term = "";
    for (let i of searchTerm) {
        term += i;
        if (counter > 0) searchTerms.push(term);
        counter += 1;
    }

    return searchTerms;
};

const handleSubmit = async (event) => {
    event.preventDefault();

    const childName = `${firstName.toUpperCase()} ${lastName.toUpperCase()}`;
    // Add the confirmation dialog
    if (!window.confirm(`Are you sure you want to add child: ${childName}?`)) {
        return;
    }

    const searchTerms = buildSearchArray(lowFirstName + " " + lowLastName);
    const child = {
        lowFirstName,
        lowLastName,
        parentName,
        dob,
        parentEmail,
        parentMobile,
        healthInfo,
```

```
        additionalInfo,
        childId: "",
        searchArray: searchTerms,
    };
    try {
        const docRef = await addDoc(collection(db, "children"), child);
        // console.log("Child added with ID: ", docRef.id);

        // Update the childId in the database
        await updateDoc(doc(db, "children", docRef.id), {
            childId: docRef.id,
        });

        setFirstName("");
        setLastName("");
        setParentName("");
        setDob("");
        setParentEmail("");
        setParentMobile("");
        setHealthInfo("");
        setAdditionalInfo("");
        setChildId(docRef.id);
    } catch (error) {
        // console.error("Error adding child: ", error);
    }
};

return (
    <>
```

```
<Navigation/>
<Container className="addChild" style={{marginTop: 10}}>
   <Card>
      <Card.Header className="formCard text-center" as="h5">
         Add Child
      </Card.Header>
      <Card.Body>
         <Form onSubmit={handleSubmit}>
            <Form.Group controlId="firstName">
               <Form.Label>First Name</Form.Label>
               <Form.Control
                  type="text"
                  placeholder="Enter first name"
                  value={firstName}
                  onChange={(event) => setFirstName(event.target.value)}
               />
            </Form.Group>

            <Form.Group controlId="lastName">
               <Form.Label>Last Name</Form.Label>
               <Form.Control
                  type="text"
                  placeholder="Enter last name"
                  value={lastName}
                  onChange={(event) => setLastName(event.target.value)}
               />
            </Form.Group>

            <Form.Group controlId="parentName">
```

```
    <Form.Label>Parent Name</Form.Label>
    <Form.Control
      type="text"
      placeholder="Enter parent name"
      value={parentName}
      onChange={(event) => setParentName(event.target.value)}
    />
  </Form.Group>

  <Form.Group controlId="dob">
    <Form.Label>Date of Birth</Form.Label>
    <Form.Control
      type="date"
      value={dob}
      onChange={(event) => setDob(event.target.value)}
    />
  </Form.Group>

  <Form.Group controlId="parentEmail">
    <Form.Label>Parent Email</Form.Label>
    <Form.Control
      type="email"
      placeholder="Enter parent email"
      value={parentEmail}
      onChange={(event) => setParentEmail(event.target.value)}
    />
  </Form.Group>

  <Form.Group controlId="parentMobile">
```

```
    <Form.Label>Parent Mobile Number</Form.Label>
    <Form.Control
      type="tel"
      placeholder="Enter parent mobile number"
      value={parentMobile}
      onChange={(event) => setParentMobile(event.target.value)}
    />
  </Form.Group>

  <Form.Group controlId="healthInfo">
    <Form.Label>Health Information</Form.Label>
    <Form.Control
      as="textarea"
      placeholder="Enter important child's health information"
      value={healthInfo}
      onChange={(event) => setHealthInfo(event.target.value)}
    />
  </Form.Group>

  <Form.Group controlId="additionalInfo">
    <Form.Label>Additional Information</Form.Label>
    <Form.Control
      as="textarea"
      placeholder="Enter additional information"
      value={additionalInfo}
      onChange={(event) => setAdditionalInfo(event.target.value)}
    />
  </Form.Group>
  <br></br>
```

```
                <Card.Footer
                  className="formCardFoot text-center"
                >
                  <Button
                    variant="primary"
                    type="submit"
                  >
                    Add Child
                  </Button>
                  <br></br>
                </Card.Footer>
              </Form>
            </Card.Body>
          </Card>
          <Container className="text-center">
            <Link to="/">
              <Button
                className="connectButton"
                variant="primary"
                size="lg"
                style={{margin: 5, backgroundColor: "#f4900c",border: "#f4900c"}}
              >
                Back to Main Menu
              </Button>
            </Link>
          </Container>
        </Container>
      </>
  );
```

```
};

export default AddChild;
```

2. Archive.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState, useEffect} from "react";
import {Container, Form, Table, Button, Card} from "react-bootstrap";
import {Link} from "react-router-dom";
import Navigation from "./Navigation";
import {
    collection,
    onSnapshot,
    query,
    where,
    orderBy,
} from "firebase/firestore";
import {db} from "../firebase";

const Archive = () => {
    const [children, setChildren] = useState([]);
    const [search, setSearch] = useState({name: "", date: ""});
    const [filteredChildren, setFilteredChildren] = useState([]);
    const [error, setError] = useState(false);

    useEffect(() => {
```

```
    const unsub = onSnapshot(collection(db, "children"), (snapshot) => {
      setChildren(snapshot.docs.map((doc) => ({...doc.data(), id: doc.id})));
    });
    return () => unsub();
  }, []);

  useEffect(() => {
    const formatDate = (dateStr) => {
      const parts = dateStr.split("/");
      if (parts.length === 3) {
        // Assuming "D/M/YYYY" format
        return `${parts[2]}-${parts[1].padStart(2, "0")}-${parts[0].padStart(
          2,
          "0"
        )}`;
      } else {
        // Assuming "YYYY-MM-DD" format
        return dateStr;
      }
    };

    const filtered = children.filter((child) => {
      const lowFirstName = child.lowFirstName.toLowerCase();
      const lowLastName = child.lowLastName.toLowerCase();
      const nameMatch = search.name
        ? lowFirstName.includes(search.name.toLowerCase()) ||
        lowLastName.includes(search.name.toLowerCase())
        : true;
      const dateMatch = search.date
```

```
            ? child.dailyReviews && child.dailyReviews.some((review) => {
                const formattedReviewDate = formatDate(review.date);
                return formattedReviewDate === search.date;
            })
            : true;

        return nameMatch && dateMatch;
    });

    // Only show the error message if there are no results and the user has entered a
name or date
    if (filtered.length === 0 && (search.name || search.date)) {
        setError(true);
        setTimeout(() => {
            setError(false);
        }, 5000);
    } else {
        setError(false);
    }

    setFilteredChildren(filtered);
}, [search, children]);

const handlePrint = () => {
    window.print();
};

return (
    <>
        <Navigation/>
```

```
<Container className="archive" style={{marginTop: 10}}>
  <Card>
    <Card.Header className="formCard text-center" as="h5">
      Archive
    </Card.Header>
    <Card.Body>
      <Form onSubmit={(e) => e.preventDefault()}>
        <Form.Group controlId="searchName">
          <Form.Label>Search by name:</Form.Label>
          <Form.Control
            type="text"
            value={search.name}
            onChange={(e) =>
              setSearch({...search, name: e.target.value})
            }
          />
          {search.name && (
            <div
              className="alert alert-danger text-center"
              role="alert"
            >
              <Button
                variant="outline-secondary"
                onClick={() => setSearch({...search, name: ""})}
              >
                Clear
              </Button>
            </div>
          )}
```

```
    </Form.Group>
    <Form.Group controlId="searchDate">
      <Form.Label>Search by date:</Form.Label>
      <Form.Control
        type="date"
        value={search.date}
        onChange={(e) =>
          setSearch({...search, date: e.target.value})
        }
      />
      {search.date && (
        <div
          className="alert alert-danger text-center"
          role="alert"
        >
          <Button
            variant="outline-secondary"
            onClick={() => setSearch({...search, date: ""})}
          >
            Clear
          </Button>
        </div>
      )}
    </Form.Group>
  </Form>
  {error && (
    <div
      className="alert alert-danger text-center"
      role="alert"
```

```
            >
              <Form.Text className="text-muted">
                <b>
                  No Review found, please check Child's Name or Date. Try again.
                </b>
              </Form.Text>
            </div>
          )}
          <Table striped bordered hover responsive="sm">
            <thead>
            <tr>
              <th>Child Name</th>
              <th>Daily Review</th>
            </tr>
            </thead>
            <tbody>
            {filteredChildren.map((child) => (
              <tr key={child.id}>
                <td>{child.lowFirstName.toUpperCase() + " " +
child.lowLastName.toUpperCase()}</td>
                <td>
                  {child.dailyReviews &&
                    child.dailyReviews.map((review, index) => (
                      <div key={index}>
                        <p>Date: {review.date}</p>
                        <p>Meal Time: {review.mealTime}</p>
                        <p>Meals: {review.meals}</p>
                        <p>Nappy Time: {review.nappyTime}</p>
                        <p>Nappy Status: {review.nappyStatus}</p>
                        <p>Activities: {review.activities}</p>
```

```
                        <p>Other Comments: {review.otherComments}</p>
                        <p>Updated By: {review.updatedBy}</p>
                        <p>
                          Updated At:{" "}
                          {new Date(
                            review.timestamp?.toDate()
                          ).toLocaleTimeString()}
                        </p>
                      </div>
                    ))}
                  </td>
                </tr>
              ))}
              </tbody>
            </Table>
            <Card.Footer
              className="formCardFoot text-center"
            >
              <Button
                onClick={handlePrint}
                variant="primary"
              >
                Print
              </Button>
              {/* Add download and copy functionality */}
              <br></br>
            </Card.Footer>
          </Card.Body>
        </Card>
```

```
            <Container className="text-center">
              <Link to="/">
                <Button
                  className="connectButton"
                  variant="primary"
                  size="lg"
                  style={{margin: 5,backgroundColor: "#f4900c",border: "#f4900c"}}
                >
                  Back to Main Menu
                </Button>
              </Link>
            </Container>
          </Container>
        </>
    );
};

export default Archive;
```

3. AuthDetails.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import {useEffect, useState, createContext} from "react";
import {auth, db} from "../firebase";
import {onAuthStateChanged} from "firebase/auth";
import {doc, getDoc} from "firebase/firestore";
```

```
export const AuthDetails = createContext();

export const AuthInfo = ({children}) => {
    const [loggedUser, setLoggedUser] = useState(null);

    useEffect(() => {
        //listening in real time
        const checkStatus = onAuthStateChanged(auth, async (user) => {
            if (user) {
                const userDoc = await getDoc(doc(db, "users", user.uid));
                if (userDoc.exists()) {
                    const {userRole, displayName, email, photoURL} = userDoc.data();
                    setLoggedUser({uid: user.uid, userRole, displayName, email, photoURL});
                }
            } else {
                setLoggedUser(null);
            }
        });
        // clean up function to prevent memory leaking
        return () => {
        };
    }, []);

    return (
        <AuthDetails.Provider value={{loggedUser}}>
            {children}
        </AuthDetails.Provider>
    );
```

```
};
```

4. Carousel.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React from 'react';
import {Carousel as BootstrapCarousel} from 'react-bootstrap';
import ChildDetails from './ChildDetails';
import {deleteDoc, doc} from 'firebase/firestore';
import {db} from "../firebase";

const Carousel = ({children, onRefresh}) => {


    const deleteChildFromDb = async (id) => {
        try {
            await deleteDoc(doc(db, "children", id));
        } catch (error) {
            // console.error("Error removing child: ", error);
        }
    };

    const updateChild = (id) => {
        // handle updating child details here
        // console.log('Update child with id:', id);
    };
```

```
    const removeChild = (id) => {
        const childToRemove = children.find((c) => c.id === id);
        const childFullName = `${childToRemove.lowFirstName}
${childToRemove.lowLastName}`.toUpperCase();

        if (window.confirm(`Are you sure you want to remove this child: ${childFullName}?`)) {
            deleteChildFromDb(id);
            // console.log("Remove child with id:", id);
            onRefresh(); // Trigger a refresh in the Home component
        }
    };
    return (
        <BootstrapCarousel interval={null}>
            {children.map((child) => (
                <BootstrapCarousel.Item key={child.id}>
                    <ChildDetails child={child} updateChild={updateChild}
removeChild={removeChild}/>
                </BootstrapCarousel.Item>
            ))}
        </BootstrapCarousel>
    );
};

export default Carousel;
```

5. Chat.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
```

```
*/
import React, {useContext} from "react";
import {Container} from "react-bootstrap";
import Messages from "./Messages";
import Input from "./Input";
import {MessageDetails} from "./MessageDetails";

const Chat = () => {
    const {data} = useContext(MessageDetails);
    return (
        <Container className="chat">
            <Container className="fixed">
                <br></br>
                <Container className="chatDetails">
                    <span>{data.user?.displayName?.toUpperCase()}</span>
                </Container>
                <br></br>
            </Container>
            <Messages/>
            <Container className="fixed-bottom">
                <Input/>
            </Container>
        </Container>
    );
};

export default Chat;
```

6. Chats.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext, useEffect, useState} from "react";
import {Container, Image} from "react-bootstrap";
import {AuthDetails} from "./AuthDetails";
import {db} from "../firebase";
import {doc, onSnapshot} from "firebase/firestore";
import {MessageDetails} from "./MessageDetails";

const Chats = () => {
    const {loggedUser} = useContext(AuthDetails);
    const {dispatch} = useContext(MessageDetails);
    const [messages, setMessages] = useState([]);

    useEffect(() => {
        const fetchMessages = async () => {
            const unsub = onSnapshot(
                doc(db, "userMessages", loggedUser.uid),
                (doc) => {
                    setMessages(doc.data() || []);
                }
            );

            return () => {
                unsub();
            };
```

```
    };
    loggedUser.uid && fetchMessages();
}, [loggedUser.uid]);
// console.log(messages);

const handleSelect = (u) => {
    dispatch({type: "CHANGE_USER", payload: u});
};

return (
    <Container className="lastMessages">
        {Object.entries(messages)
            ?.sort((a, b) => b[1].date - a[1].date)
            .map((mess) => (
                <Container
                    className="loggedUserMessages"
                    key={mess[0]}
                    onClick={() => handleSelect(mess[1].userInfo)}
                >
                    <Image
                        className="avatar"
                        src={mess[1].userInfo.photoURL}
                        width="30"
                        height="30"
                        alt="avatar"
                        roundedCircle
                    />
                    <Container className="userInfo">
                        <span>{mess[1].userInfo.displayName.toUpperCase()}</span>
```

```
                    <p>{mess[1].lastMessage?.text}</p>
                    {/*{console.log(mess[1].userInfo.displayName)}*/}
                    {/*{console.log(mess[1].lastMessage?.text)}*/}
                </Container>
            </Container>
        ))}
    </Container>
  );
};

export default Chats;
```

7. ChildDetails.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext, useState} from "react";
import {Card, Button} from "react-bootstrap";
import {Link} from "react-router-dom";
import {AuthDetails} from "./AuthDetails";

const ChildDetails = ({child, removeChild}) => {
  let fullName = `${child.lowFirstName} ${child.lowLastName}`; // combine first and last name
  let fullNameCaps = fullName.toUpperCase(); // convert to uppercase

  const [showCopyPopup, setShowCopyPopup] = useState(false);
  const {loggedUser} = useContext(AuthDetails);
```

```
const getCurrentFormattedDate = () => {
    const today = new Date();
    return `${today.getDate()}/${today.getMonth() + 1}/${today.getFullYear()}`;
};

const copyDailyReviews = () => {
    const reviews = child.dailyReviews
        .filter((review) => review.date === getCurrentFormattedDate())
        .map((review, index) => {
            return `
  Meal Time: ${review.mealTime}
  Meals: ${review.meals}
  Nappy Time: ${review.nappyTime}
  Nappy Status: ${review.nappyStatus}
  Activities: ${review.activities}
  Other Comments: ${review.otherComments}
  Updated By: ${review.updatedBy}
  Updated At: ${new Date(review.timestamp?.toDate()).toLocaleTimeString()}
        `;
        });

    const reviewsText = reviews.join("\n\n");

    navigator.clipboard.writeText(reviewsText).then(
        () => {
            // console.log("Daily reviews copied to clipboard!");
            setShowCopyPopup(true);
            setTimeout(() => {
```

```
            setShowCopyPopup(false);
          }, 3000);
        },
        (err) => {
          // console.error("Could not copy daily reviews: ", err);
        }
      );
  };

  return (
    <Card className="carousel">
      <Card.Header className="formCard text-center" as="h5">
        {fullNameCaps}
      </Card.Header>
      <Card.Body>
        <Card.Text>
          <b>Date of Birth:</b> <br></br>
          {child.dob}
        </Card.Text>
        <Card.Text>
          <b>Parent Name:</b> <br></br>
          {child.parentName}
        </Card.Text>
        <Card.Text>
          <b>Parent Email:</b> <br></br>
          {child.parentEmail}
        </Card.Text>
        <Card.Text>
          <b>Parent Mobile Number:</b> <br></br>
```

```
    {child.parentMobile}
</Card.Text>
<Card.Text>
    <b>Child's Health Info:</b> <br></br>
    {child.healthInfo}
</Card.Text>
<Card.Text>
    <b>Additional Information:</b> <br></br>
    {child.additionalInfo}
</Card.Text>
<Card.Text>
    <b><u>DAILY REVIEWS - {new Date().toLocaleDateString()}</u></b>
    {child.dailyReviews &&
        child.dailyReviews
            .filter((review) => review.date === getCurrentFormattedDate())
            .map((review, index) => (
                <div key={index} style={{marginBottom: "10px"}}>
                    <b>Meal Time:</b> {review.mealTime} <br/>
                    <b>Meals:</b> {review.meals} <br/>
                    <b>Nappy Time:</b> {review.nappyTime} <br/>
                    <b>Nappy Status:</b> {review.nappyStatus} <br/>
                    <b>Activities:</b> {review.activities} <br/>
                    <b>Other Comments:</b> {review.otherComments} <br/>
                    <b>Updated By:</b> {review.updatedBy} <br/>
                    <b>Updated At:</b>{" "}
                    {new Date(review.timestamp?.toDate()).toLocaleTimeString()}
                </div>
            ))}
</Card.Text>
```

```
<Card.Footer
  className="formCardFoot text-center"
>
  <Link to={`/daily-review/${child.id}`}>
    <Button
      variant="primary"
    >
      Add Review
    </Button>
  </Link>
  <Button
    variant="primary"
    onClick={copyDailyReviews}
  >
    Copy Review
  </Button>
  {loggedUser && loggedUser.userRole === "manager" && (
    <Link to={`/update/${child.id}`}>
      <Button
        variant="primary"
      >
        Update Child
      </Button>
    </Link>
  )}
  {loggedUser && loggedUser.userRole === "manager" && (
    <Button
      variant="danger"
      style={{
```

```
                backgroundColor: "crimson",
                border: "crimson",
              }}
              onClick={() => removeChild(child.id)}
            >
              Remove
            </Button>
          )}
          <br></br>
          {showCopyPopup && (
            <div
              style={{
                position: "fixed",
                top: "50%",
                left: "50%",
                transform: "translate(-50%, -50%)",
                zIndex: 100,
                backgroundColor: "#f4900c",
                borderRadius: "5px",
                padding: "10px",
                boxShadow: "0 0 10px rgba(0, 0, 0, 0.2)",
              }}
            >
              Daily reviews copied to clipboard!
            </div>
          )}
        </Card.Footer>
      </Card.Body>
    </Card>
```

```
    );
};

export default ChildDetails;
```

8. DailyReview.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState, useContext, useEffect} from "react";
import {useParams, Link, useNavigate} from "react-router-dom";
import {Form, Button, Card, Container} from "react-bootstrap";
import Navigation from "./Navigation";
import {AuthDetails} from "./AuthDetails";
import {doc, updateDoc, arrayUnion, getDoc} from "firebase/firestore";
import {db} from "../firebase";

const DailyReview = () => {
    const [date, setDate] = useState("");
    const [mealTime, setMealTime] = useState("");
    const [meals, setMeals] = useState("");
    const [nappyTime, setNappyTime] = useState("");
    const [nappyStatus, setNappyStatus] = useState("");
    const [activities, setActivities] = useState("");
    const [otherComments, setOtherComments] = useState("");
    const navigate = useNavigate();
    const {loggedUser} = useContext(AuthDetails);
    const [childName, setChildName] = useState("");
```

```
  const {id} = useParams(); // Get the child ID from the URL

  const today = new Date();
  const formattedDate = `${today.getDate()}/${today.getMonth() +
1}/${today.getFullYear()}`;

  // Fetch the child data from the database based on the child ID
  useEffect(() => {
    const fetchChild = async () => {
      try {
        const docRef = doc(db, "children", id);
        const docSnap = await getDoc(docRef);

        if (docSnap.exists()) {
          setChildName({id: docSnap.id, ...docSnap.data()});
        } else {
          // console.log("No such document!");
        }
      } catch (error) {
        // console.error("Error fetching child: ", error);
      }
    };

    fetchChild();
  }, [id]);

  const handleSubmit = async (e) => {
    e.preventDefault();
```

```
    if (!window.confirm(`Are you happy with the review information for child:
${childName.lowFirstName.toUpperCase()} ${childName.lowLastName.toUpperCase()}?`))
return;

    const newReview = {
      date: formattedDate,
      mealTime,
      meals,
      nappyTime,
      nappyStatus,
      activities,
      otherComments,
      updatedBy: loggedUser.displayName.toUpperCase(),
      timestamp: new Date(),
    };

    try {
      const childDoc = doc(db, "children", id);
      await updateDoc(childDoc, {
        dailyReviews: arrayUnion(newReview),
      });
      navigate(`/`);
    } catch (error) {
      // console.error("Error adding daily review:", error);
    }
  };

  return (
    <>
      <Navigation/>
```

```
<Container className="dailyReview" style={{marginTop: 10}}>
  <Card>
    <Card.Header className="formCard text-center" as="h5">
      Daily Review - {new Date().toLocaleDateString()}
    </Card.Header>
    <Card.Body>
      <Form onSubmit={handleSubmit}>
        <Form.Group controlId="meals">
          <Form.Label>Meals</Form.Label>
          <Form.Control
            type="time"
            name="mealTime"
            value={mealTime}
            onChange={(event) => setMealTime(event.target.value)}
          />
          <Form.Control
            as="textarea"
            name="meals"
            placeholder="Enter what the child ate"
            value={meals}
            onChange={(event) => setMeals(event.target.value)}
          />
        </Form.Group>

        <Form.Group controlId="nappies">
          <Form.Label>Nappies</Form.Label>
          <Form.Control
            type="time"
            name="nappyTime"
```

```
        value={nappyTime}
        onChange={(event) => setNappyTime(event.target.value)}
      />
      <Form.Select
        name="nappyStatus"
        value={nappyStatus}
        onChange={(event) => setNappyStatus(event.target.value)}
      >
        <option value="">Select Nappy Status</option>
        <option value="wet">Wet</option>
        <option value="soiled">Soiled</option>
        <option value="dry">Dry</option>
      </Form.Select>
    </Form.Group>

    <Form.Group controlId="activities">
      <Form.Label>Activities</Form.Label>
      <Form.Control
        as="textarea"
        name="activities"
        placeholder="Enter activities"
        value={activities}
        onChange={(event) => setActivities(event.target.value)}
      />
    </Form.Group>

    <Form.Group controlId="otherComments">
      <Form.Label>Other Comments</Form.Label>
      <Form.Control
```

```
        as="textarea"
        name="otherComments"
        placeholder="Enter other comments"
        value={otherComments}
        onChange={(event) => setOtherComments(event.target.value)}
      />
    </Form.Group>

    <Form.Group controlId="updatedBy">
      <Form.Label>Updated By</Form.Label>
      <Form.Control
        type="text"
        name="updatedBy"
        readOnly
        value={loggedUser.displayName.toUpperCase()}
      />
    </Form.Group>
    <br></br>
    <Card.Footer
      className="formCardFoot text-center"
    >
      <Button
        variant="primary"
        type="submit"
      >
        Add Review
      </Button>
      <br></br>
    </Card.Footer>
```

```
                </Form>
              </Card.Body>
            </Card>
            <Container className="text-center">
              <Link to="/">
                <Button
                  className="connectButton"
                  variant="primary"
                  size="lg"
                  style={{margin: 5,backgroundColor: "#f4900c",border: "#f4900c"}}
                >
                  Back to Main Menu
                </Button>
              </Link>
            </Container>
          </Container>
        </>
    );
};

export default DailyReview;
```

9. ForgotPassword.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState} from "react";
import {auth} from "../firebase";
```

```
import {sendPasswordResetEmail} from "firebase/auth";
import {Container, Form, Button, Card, Image} from "react-bootstrap";
import {useNavigate, Link} from "react-router-dom";

const ForgotPassword = () => {
    const [error, setError] = useState(false);
    const [resetEmailSent, setResetEmailSent] = useState(false);
    const navigate = useNavigate();

    const handleForgotPassword = async (event) => {
        event.preventDefault();
        const email = event.target[0].value;

        if (email) {
            try {
                await sendPasswordResetEmail(auth, email);
                setResetEmailSent(true);
                setTimeout(() => {
                    navigate("/login"); // or use "/" if login is the root path
                }, 3000); // Redirects after 3 seconds
            } catch (error) {
                // console.error("Error sending password reset email:", error);
                // console.error("Error code:", error.code);
                // console.error("Error message:", error.message);

                if (error.code === "auth/user-not-found") {
                    setError("No user found with this email address.");
                } else {
                    setError("An error occurred. Please try again.");
```

```
            }
        }
    } else {
        setError(true);
    }
};

return (
    <div>
        <Container className="form forgotPass" style={{marginTop: 10}}>
            <Card>
                <Card.Header className="formCard text-center" as="h5">
                    <Image src="/logo.png" alt="logo" rounded/>
                    Forgot Password
                </Card.Header>
                <Card.Body>
                    <Form onSubmit={handleForgotPassword} inline>
                        <Form.Group className="mb-3" controlId="formBasicEmail">
                            <Form.Control required type="email"
                                          placeholder="Enter email registered with the application."/>
                        </Form.Group>
                        {resetEmailSent && (
                            <div className="alert alert-success" role="alert">
                                <Form.Text className="text-muted text-center">
                                    <b>Password reset email sent! Please check your inbox.</b>
                                </Form.Text>
                            </div>
                        )}
                        {error && (
```

```
                    <div className="alert alert-danger text-center" role="alert">
                      <Form.Text className="text-muted">
                        <b>{error}</b>
                      </Form.Text>
                    </div>
                  )}
                  <Container className="text-center">
                    <Button
                      variant="primary"
                      type="submit"
                      size="lg"
                      style={{margin:5, backgroundColor: "#f4900c",border: "#f4900c"}}
                    >
                      Send Reset Email
                    </Button>
                  </Container>
                </Form>
              </Card.Body>
              <Card.Footer className="formCardFoot text-center">
                Go back to{" "}
                <Link to="/login">
                  <b>Login</b>
                </Link>
              </Card.Footer>
            </Card>
          </Container>
        </div>
    );
};
```

```
export default ForgotPassword;
```

10. Input.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext, useState} from "react";
import {Container, Form, Button} from "react-bootstrap";
import {BiSend} from "react-icons/bi";
import {CgAttachment} from "react-icons/cg";
import {AuthDetails} from "./AuthDetails";
import {MessageDetails} from "./MessageDetails";
import {db, storage} from "../firebase";
import {getDownloadURL, ref, uploadBytesResumable} from "firebase/storage";
import {v4 as uuidv4} from "uuid";
import {
    doc,
    Timestamp,
    updateDoc,
    arrayUnion,
    serverTimestamp,
} from "firebase/firestore";

const Input = () => {
    const {loggedUser} = useContext(AuthDetails);
    const {data} = useContext(MessageDetails);
```

```
const [text, setText] = useState("");
const [image, setImage] = useState(null);
const [err, setErr] = useState(false);
const [errorMessage, setErrorMessage] = useState("");
const [showErrorMessage, setShowErrorMessage] = useState(false);

const handleSend = async (event) => {
    event.preventDefault();

    // Check if both text and image are empty, then return and do nothing
    if (!text.trim() && !image) {
        return;
    }

    // Clear the input, image state, and any previous error messages
    setText("");
    setImage(null);
    setErrorMessage("");

    try {
        if (image) {
            const storageRef = ref(storage, uuidv4());

            const uploadTask = uploadBytesResumable(storageRef, image);

            uploadTask.on(
                (_err) => {
                    //TODO:Handle Error
                    setErr(true);
```

```
        },
        () => {
          getDownloadURL(uploadTask.snapshot.ref).then(async (downloadURL) => {
            await updateDoc(doc(db, "allMessages", data.messageId), {
              messages: arrayUnion({
                id: uuidv4(),
                text,
                senderId: loggedUser.uid,
                date: Timestamp.now(),
                img: downloadURL,
              }),
            });
          });
        }
      );
    } else {
      await updateDoc(doc(db, "allMessages", data.messageId), {
        messages: arrayUnion({
          id: uuidv4(),
          text,
          senderId: loggedUser.uid,
          date: Timestamp.now(),
        }),
      });
    }

    await updateDoc(doc(db, "userMessages", loggedUser.uid), {
      [data.messageId + ".lastMessage"]: {
        text,
```

```
            },
            [data.messageId + ".date"]: serverTimestamp(),
        });

        await updateDoc(doc(db, "userMessages", data.user.uid), {
            [data.messageId + ".lastMessage"]: {
                text,
            },
            [data.messageId + ".date"]: serverTimestamp(),
        });
    } catch (error) {
        // console.error("Error sending message:", error);
        setErrorMessage("An error occurred while sending the message.");
        setShowErrorMessage(true);

        // Remove the error message after 3 seconds
        setTimeout(() => {
            setShowErrorMessage(false);
            setErrorMessage("");
        }, 3000);

        // Optionally, restore the input text and image to their previous values
        setText(event.target.value);
        setImage(event.target.files[0]);
    }
};
return (
    <Container className="input">
        <Form.Control
```

```
            className="inputText"
            type="text"
            placeholder="Message..."
            onChange={(event) => setText(event.target.value)}
            value={text}
          />
          <Form.Group className="inputArea" controlId="formFile">
            <Form.Label>
              <CgAttachment/>
            </Form.Label>
            <Form.Control
              type="file"
              style={{display: "none"}}
              onChange={(event) => setImage(event.target.files[0])}
            />
          </Form.Group>
          {errorMessage && (
            <Form.Text className="text-danger">
              <b>{errorMessage}</b>
            </Form.Text>
          )}
          <Button variant="custSend" onClick={handleSend}>
            <BiSend/>
          </Button>
        </Container>
    );
};

export default Input;
```

11. MainMenu.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext} from "react";
import {Container, Button, Row, Col} from "react-bootstrap";
import {Link} from "react-router-dom";
import {AuthDetails} from "./AuthDetails";

function MainMenu() {
    const {loggedUser} = useContext(AuthDetails);
    return (
        <Container className="mainmenu" style={{margin: 5}}>
            <Row className="text-center">
                <Col md={{span: 6, offset: 3}}>
                    {loggedUser && (loggedUser.userRole === "manager") && (
                        <Link to="/addchild">
                            <Button className="addChildButton" variant="primary" size="lg"
style={{margin: 5}}>
                                Add a child
                            </Button>
                        </Link>
                    )}

                    <Link to="/connect">
                        <Button className="connectButton" variant="primary" size="lg"
style={{margin: 5}}>
```

```
                    Message
                </Button>
            </Link>

            <Link to="/archive">
                <Button className="archiveButton" variant="primary" size="lg"
style={{margin: 5}}>
                    Archive
                </Button>
            </Link>
        </Col>
    </Row>
</Container>
);
}

export default MainMenu;
```

12. Message.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext, useEffect, useRef} from "react";
import {Container, Image} from "react-bootstrap";
import {AuthDetails} from "./AuthDetails";
import {MessageDetails} from "./MessageDetails";

const Message = ({message}) => {
```

```
const {loggedUser} = useContext(AuthDetails);
const {data} = useContext(MessageDetails);
// console.log(message);

const ref = useRef();

useEffect(() => {
   ref.current?.scrollIntoView({behavior: "smooth"});
}, [message]);

const isLoggedUser = message.senderId === loggedUser.uid;

return (
   <>
      <Container
         ref={ref}
         className={`message ${isLoggedUser ? "loggedUser" : "recipient"}`}
      >
         {isLoggedUser || <Image
            className="avatar"
            src={data.user.photoURL}
            alt="avatar"
            roundedCircle
         />}
         <Container className="messageSubject">
            <p>{message.text}</p>
            {message.img && <Image src={message.img} alt=""/>}
         </Container>
         {isLoggedUser && <Image
```

```
                    className="avatar"
                    src={loggedUser.photoURL}
                    alt="avatar"
                    roundedCircle
                />}
            </Container>
        </>
    );
};

export default Message;
```

13. MessageDetails.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import {createContext, useContext, useReducer} from "react";
import {AuthDetails} from "./AuthDetails";

export const MessageDetails = createContext();

export const MessageInfo = ({children}) => {
    const {loggedUser} = useContext(AuthDetails);
    const INITIAL_STATE = {
        messageId: "null",
        user: {},
    };
```

```
  const messageReducer = (state, action) => {
    switch (action.type) {
      case "CHANGE_USER":
        return {
          user: action.payload,
          messageId:
            loggedUser.uid > action.payload.uid
              ? loggedUser.uid + action.payload.uid
              : action.payload.uid + loggedUser.uid,
        };

      default:
        return state;
    }
  };

  const [state, dispatch] = useReducer(messageReducer, INITIAL_STATE);

  return (
    <MessageDetails.Provider value={{data: state, dispatch}}>
      {children}
    </MessageDetails.Provider>
  );
};
```

14. Messages.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
```

```
*/
import React, {useContext, useEffect, useState} from "react";
import {Container} from "react-bootstrap";
import Message from "./Message";
import {MessageDetails} from "./MessageDetails";
import {doc, onSnapshot} from "firebase/firestore";
import {db} from "../firebase";

const Messages = () => {
    const {data} = useContext(MessageDetails);
    const [messages, setMessages] = useState([]);

    useEffect(() => {
        const unsub = onSnapshot(doc(db, "allMessages", data.messageId), (doc) => {
            if (doc.exists()) {

                setMessages(doc.data().messages);
            } else {
                // console.log("No such document!");
            }
        });
        return () => unsub();
    }, [data.messageId]);

    return (
        <Container className="messages">
            {messages &&
                messages.map((message) => (
                    <Message message={message} key={message.id}/>
```

```
            ))}
        </Container>
    );
};

export default Messages;
```

15. Navigation.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import {Navbar, Nav, Container, Button, Image, Form} from "react-bootstrap";
import React, {useContext} from "react";
import {signOut} from "firebase/auth";
import {AuthDetails} from "./AuthDetails";
import {auth} from "../firebase";

const Navigation = () => {
    const {loggedUser} = useContext(AuthDetails);
    // console.log({ loggedUser });

    return (
        <Navbar
            className="navigation"
            collapseOnSelect
            expand="lg"
            bg="dark"
            variant="dark"
```

```
      >
        <Container>
          <Image src="/logo.png" alt="logo" rounded/>
          <Navbar.Brand href="#home">Creche Connect</Navbar.Brand>
          <Navbar.Toggle aria-controls="responsive-navbar-nav"/>
          <Navbar.Collapse id="responsive-navbar-nav">
            <Nav className="me-auto"></Nav>
            <Nav>
              {loggedUser && (
                <>
                  <Form.Text className="text-muted text-center" style={{padding: 10}}>
                    {loggedUser.displayName.toUpperCase()}
                  </Form.Text>
                  <Button
                    className="logoutButton"
                    variant="light"
                    onClick={() => signOut(auth)}
                  >
                    Logout
                  </Button>
                </>
              )}
            </Nav>
          </Navbar.Collapse>
        </Container>
      </Navbar>
    );
};
```

```
export default Navigation;
```

16. Search.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext, useState, useEffect} from "react";
import {Container, Form, Image} from "react-bootstrap";
import {
    collection,
    query,
    where,
    getDocs,
    setDoc,
    doc,
    updateDoc,
    serverTimestamp,
    getDoc,
} from "firebase/firestore";
import {db} from "../firebase";
import {AuthDetails} from "./AuthDetails";

const Search = () => {
    const [userName, setUserName] = useState("");
    const [user, setUser] = useState(null);
    const [err, setErr] = useState(false);
    const [showError, setShowError] = useState(false);
    const [filteredUsers, setFilteredUsers] = useState([]);
```

```
const {loggedUser} = useContext(AuthDetails);

const handleSearch = async () => {
    const q = query(
        collection(db, "users"),
        where("searchArray", "array-contains", userName.toLowerCase())
    );

    try {
        const querySnapshot = await getDocs(q);
        if (!querySnapshot.empty) {
            const usersArray = querySnapshot.docs
                .map((doc) => doc.data())
                .filter((user) => user.uid !== loggedUser.uid);
            setUser(usersArray);
            setErr(false); // Reset error state if user is found
        } else {
            setUser(null);
            setErr(true);
            setShowError(true); // Set showError to true when person is not found

            setTimeout(() => {
                setShowError(false); // Reset showError to false after 3 seconds
            }, 3000);
        }
    } catch (err) {
        // console.error("Error searching for user:", err);
        setUser(null);
```

```
        setErr(true);
        setShowError(true); // Set showError to true when error occurs

        setTimeout(() => {
            setShowError(false); // Reset showError to false after 3 seconds
        }, 3000);
    }
};



const handleKey = (event) => {
    event.key === "Enter" && handleSearch();
};

const handleMessaging = async (user) => {
    // verify if person messages exist in database(create if not)
    const loggedUserRef =
        loggedUser.uid > user.uid
            ? loggedUser.uid + user.uid
            : user.uid + loggedUser.uid;
    try {
        const response = await getDoc(doc(db, "allMessages", loggedUserRef));
        //create message in database
        if (!response.exists()) {
            //create a chat in allMessages collection
            await setDoc(doc(db, "allMessages", loggedUserRef), {
                messageDetails: [],
            });
        }
```

```
        //create person messages
        await updateDoc(doc(db, "userMessages", loggedUser.uid), {
            [loggedUserRef + ".userInfo"]: {
                uid: user.uid,
                displayName: user.displayName,
                photoURL: user.photoURL,
            },
            [loggedUserRef + ".date"]: serverTimestamp(),
        });

        await updateDoc(doc(db, "userMessages", user.uid), {
            [loggedUserRef + ".userInfo"]: {
                uid: loggedUser.uid,
                displayName: loggedUser.displayName,
                photoURL: loggedUser.photoURL,
            },
            [loggedUserRef + ".date"]: serverTimestamp(),
        });
    } catch (err) {
        setErr(true);
    }

    setUser(null);
    setUserName("");
};

useEffect(() => {
    const filterUsers = () => {
        const filtered = user
```

```
            ? user.filter((u) => {
                const displayName = u.displayName.toLowerCase();
                const nameMatch = userName
                  ? displayName.includes(userName.toLowerCase())
                  : true;
                return nameMatch;
              })
            : [];
          setFilteredUsers(filtered);
        };
        filterUsers();
      }, [userName, user]);

      return (
        <Container className="search">
          <Container className="searchInput">
            <Form.Control
              className="inputSearch"
              type="text"
              placeholder="Find a Person"
              onKeyDown={handleKey}
              onChange={(event) => setUserName(event.target.value)}
              value={userName}
            />
          </Container>
          {showError && (
            <Form.Text className="text-muted text-center">
              <b>Person not found</b>
            </Form.Text>
```

```
        )}
        {filteredUsers && (
          filteredUsers.map((u) => (
            <Container
              key={u.uid}
              className="userMessages"
              onClick={() => handleMessaging(u)}
              style={{marginTop: 10}}>
              <Image
                className="avatar"
                src={u.photoURL}
                alt="avatar"
                width="30"
                height="30"
                roundedCircle
              />
              <Container className="userMessagesInfo">
                <span>{u.displayName.toUpperCase()}</span>
              </Container>
            </Container>
          )
        ))}
      </Container>
    );
};

export default Search;
```

17. Sidebar.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React from "react";
import {Container} from "react-bootstrap";
import Search from "./Search";
import Chats from "./Chats";

const Sidebar = () => {
    return (
        <>
            <Container className="sidebar">
                <Container className="fixed">
                    <br></br>
                    <Search/>
                    <br></br>
                    <Container className="text-center">
                        <span style={{color: "#273c4d", height: 20, display: "inline-block",
marginBottom: 15}}>Your last messages</span>
                    </Container>
                </Container>
                <Chats/>
            </Container>
        </>
    );
};

export default Sidebar;
```

18. UpdateChild.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState, useEffect} from "react";
import {Form, Button, Card, Container} from "react-bootstrap";
import {useNavigate, useParams, Link} from "react-router-dom";
import {doc, getDoc} from "firebase/firestore";
import {db} from "../firebase";
import Navigation from "./Navigation";

const UpdateChild = ({onUpdate}) => {
    const [updatedChild, setUpdatedChild] = useState({});
    const navigate = useNavigate();
    const {id} = useParams(); // Get the child ID from the URL

    // Fetch the child data from the database based on the child ID
    useEffect(() => {
        const fetchChild = async () => {
            try {
                const docRef = doc(db, "children", id);
                const docSnap = await getDoc(docRef);

                if (docSnap.exists()) {
                    setUpdatedChild({id: docSnap.id, ...docSnap.data()});
                } else {
                    // console.log("No such document!");
                }
```

```
        } catch (error) {
            // console.error("Error fetching child: ", error);
        }
    };

    fetchChild();
}, [id]);

const handleSubmit = async (e) => {
    e.preventDefault();

    if (
        window.confirm(
            `Are you sure you want to update child:
${updatedChild.lowFirstName.toUpperCase()}
${updatedChild.lowLastName.toUpperCase()}?`
        )
    ) {
        await onUpdate(updatedChild);
        navigate("/");
    }
};

const handleChange = (event) => {
    const {name, value} = event.target;
    setUpdatedChild({...updatedChild, [name]: value});
};

return (
    <>
```

```
<Navigation/>
<Container className="updateChild" style={{marginTop: 10}}>
   <Card>
      <Card.Header className="formCard text-center" as="h5">
         Update Child
      </Card.Header>
      <Card.Body>
         <Form onSubmit={handleSubmit}>
            <Form.Group controlId="firstName">
               <Form.Label>First Name</Form.Label>
               <Form.Control
                  type="text"
                  name="lowFirstName"
                  value={updatedChild.lowFirstName || ""}
                  onChange={handleChange}
               />
            </Form.Group>

            <Form.Group controlId="lastName">
               <Form.Label>Last Name</Form.Label>
               <Form.Control
                  type="text"
                  name="lowLastName"
                  value={updatedChild.lowLastName || ""}
                  onChange={handleChange}
               />
            </Form.Group>

            <Form.Group controlId="parentName">
```

```
    <Form.Label>Parent Name</Form.Label>
    <Form.Control
      type="text"
      name="parentName"
      value={updatedChild.parentName || ""}
      onChange={handleChange}
    />
  </Form.Group>

  <Form.Group controlId="dob">
    <Form.Label>Date of Birth</Form.Label>
    <Form.Control
      type="date"
      name="dob"
      value={updatedChild.dob || ""}
      onChange={handleChange}
    />
  </Form.Group>

  <Form.Group controlId="parentEmail">
    <Form.Label>Parent Email</Form.Label>
    <Form.Control
      type="email"
      name="parentEmail"
      value={updatedChild.parentEmail || ""}
      onChange={handleChange}
    />
  </Form.Group>
```

```
<Form.Group controlId="parentMobile">
  <Form.Label>Parent Mobile Number</Form.Label>
  <Form.Control
    type="tel"
    name="parentMobile"
    value={updatedChild.parentMobile || ""}
    onChange={handleChange}
  />
</Form.Group>

<Form.Group controlId="healthInfo">
  <Form.Label>Health Information</Form.Label>
  <Form.Control
    as="textarea"
    name="healthInfo"
    value={updatedChild.healthInfo || ""}
    onChange={handleChange}
  />
</Form.Group>

<Form.Group controlId="additionalInfo">
  <Form.Label>Additional Information</Form.Label>
  <Form.Control
    as="textarea"
    name="additionalInfo"
    value={updatedChild.additionalInfo || ""}
    onChange={handleChange}
  />
</Form.Group>
```

```
                    <br></br>
                    <Card.Footer className="formCardFoot text-center">
                      <Button
                        variant="primary"
                        type="submit">
                        Save Changes
                      </Button>
                      <br></br>
                    </Card.Footer>
                  </Form>
                </Card.Body>
              </Card>
              <Container className="text-center">
                <Link to="/">
                  <Button
                    className="connectButton" variant="primary" size="lg" style={{margin: 5,
backgroundColor: "#f4900c",border: "#f4900c" }}>
                    Back to Main Menu
                  </Button>
                </Link>
              </Container>
            </Container>
        </>
    );
};

export default UpdateChild;
```

19. Welcome.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext} from "react";
import {Container, Row, Col} from "react-bootstrap";
import {AuthDetails} from "./AuthDetails";

function Welcome() {
    const {loggedUser} = useContext(AuthDetails);

    return (
        <Container className="welcome">
            <Container>
                <Row className="text-center">
                    <Col md={{span: 6, offset: 3}}>
                        <h1>Welcome, <br></br> {loggedUser.displayName.toUpperCase()}</h1>
                    </Col>
                    <q>
                        <b>
                            We enhance the quality of a child's development by improving
                            communication between childcare practitioners and parents.
                        </b>
                    </q>
                </Row>
            </Container>
        </Container>
    );
}
```

```
export default Welcome;
```

20. Connect.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useContext} from "react";
import {Container, Button, Card, CardGroup} from "react-bootstrap";
import Sidebar from "../components/Sidebar";
import Chat from "../components/Chat";
import Navigation from "../components/Navigation";
import {Link} from "react-router-dom";
import {AuthDetails} from "../components/AuthDetails";


function Connect() {
    const {loggedUser} = useContext(AuthDetails);

    // console.log("user", loggedUser);

    return (
        <>
            <Navigation/>
            <Container className="connect">
                <Container className="wrapper">
                    <CardGroup className="connectArea">
                        <Card className="sidebarCard">
```

```
                    <Sidebar/>
                </Card>
                <Card className="chatCard">
                    <Chat/>
                </Card>
            </CardGroup>
            {loggedUser && (loggedUser.userRole === "staff" || loggedUser.userRole === "manager") && (
                <Container className="text-center">
                    <Link to="/">
                        <Button className="connectButton" variant="primary" size="lg" style={{margin: 5,backgroundColor: "#f4900c", border: "#f4900c"}}>
                            Back to Main Menu
                        </Button>
                    </Link>
                </Container>
            )}
        </Container>
    </Container>
  </>
  );
}

export default Connect;
```

21. Home.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
```

```
*/
import React, {useContext, useState, useEffect} from "react";
import {Container, Form} from "react-bootstrap";
import MainMenu from "../components/MainMenu";
import Navigation from "../components/Navigation";
import Welcome from "../components/Welcome";
import {AuthDetails} from "../components/AuthDetails";
import {collection, getDocs} from "firebase/firestore";
import {db} from "../firebase";
import Carousel from "../components/Carousel";

function Home() {
    const {loggedUser} = useContext(AuthDetails);
    const [children, setChildren] = useState([]);
    const [searchTerm, setSearchTerm] = useState("");
    const [refresh, setRefresh] = useState(false);

    const handleRefresh = () => {
        setRefresh(!refresh);
    };

    useEffect(() => {
        const fetchChildren = async () => {
            const querySnapshot = await getDocs(collection(db, "children"));
            const data = querySnapshot.docs.map((doc) => ({id: doc.id, ...doc.data()}));
            setChildren(data);
            // console.log(data); // Log the fetched data to the console
        };
```

```
    fetchChildren();
}, [loggedUser, refresh]);

const handleSearch = (event) => {
    setSearchTerm(event.target.value);
};

const filteredChildren = children.filter((child) => {
    const lowFirstName = child.lowFirstName || "";
    const lowLastName = child.lowLastName || "";

    return (
        lowFirstName.includes(searchTerm.toLowerCase()) ||
        lowLastName.includes(searchTerm.toLowerCase())
    );
});

return (
    <>
        <Navigation/>
        <Container className='home'>
            <Welcome/>
            <Container>
                <Form.Control
                    type="text"
                    placeholder="Search for a child by name"
                    onChange={handleSearch}
                    value={searchTerm}
                    className="mb-3"
```

```
                    style={{marginTop: "10px"}}
                />
                {filteredChildren.length > 0 ? (
                    <Carousel key={filteredChildren.length} children={filteredChildren}
onRefresh={handleRefresh}/>
                ) : (
                    <p style={{textAlign: 'center', marginTop: '20px'}}>
                        {searchTerm
                            ? "There is no child with this name in the system."
                            : "No children found."}
                    </p>
                )}
            </Container>
            <MainMenu/>
        </Container>
    </>
  );
}

export default Home;
```

22. Login.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState} from "react";
import {Form, Button, Container, Card, Image} from "react-bootstrap";
import {auth} from "../firebase";
```

```
import {signInWithEmailAndPassword} from "firebase/auth";
import {Link, useNavigate} from "react-router-dom";
import validator from "validator";
import DOMPurify from "dompurify";

const Login = () => {
    const navigate = useNavigate();
    const [error, setError] = useState(false);

    const handleSubmit = async (event) => {
        event.preventDefault();
        const email = event.target[0].value;
        const password = event.target[1].value;

        // Validate the email address
        if (!validator.isEmail(email)) {
            setError("Please enter a valid email address.");
            return;
        }

        // Sanitize the email address and password
        const sanitizedEmail = DOMPurify.sanitize(email);
        const sanitizedPassword = DOMPurify.sanitize(password);

        try {
            await signInWithEmailAndPassword(auth, sanitizedEmail, sanitizedPassword);
            navigate("/");

        } catch (error) {
```

```
        // Handle error
        setError(true);
    }
};

return (
    <Container className="form login">
        <Card>
            <Card.Header className="formCard text-center" as="h5">
                <Image src="/logo.png" alt="logo" rounded/>
                Login
            </Card.Header>
            <Card.Body>
                <Form onSubmit={handleSubmit}>
                    <Form.Group className="mb-3" controlId="formBasicEmail">
                        <Form.Label>Email address</Form.Label>
                        <Form.Control
                            required
                            type="email"
                            placeholder="email"
                        />
                        <Form.Control.Feedback>that is incorrect</Form.Control.Feedback>
                        <Form.Text className="text-muted">
                            We'll never share your email with anyone else.
                        </Form.Text>
                    </Form.Group>
                    <Form.Group className="mb-3" controlId="validationCustom03">
                        <Form.Label>Password</Form.Label>
                        <Form.Control
```

```
                    type="password"
                    placeholder="password"
                  />
                  <Form.Control.Feedback>Looks ok!</Form.Control.Feedback>

                  <Form.Text className="text-muted">
                    Your password must be minimum 8 characters long, contain
                    letters, numbers, special characters, upper and lower cases.
                  </Form.Text>
                </Form.Group>
                <Container className="text-center">
                  <Button variant="primary" type="submit" size="lg" style={{margin: 5,
backgroundColor: "#f4900c",border: "#f4900c"}}>
                    Sign In
                  </Button>
                </Container>
              </Form>
              {error && <div className="alert alert-danger text-center"
role="alert"><Form.Text
                classNamе="text-muted"><b>Entered details need to be corrected. Try
again.</b></Form.Text>
              </div>}
            </Card.Body>
            <Card.Footer className="formCardFoot text-center">
              Did you <Link to="/forgotpassword"><b>Forgot
Password?</b></Link><br></br>Are you not
              registered? <Link to="/register"><b>Register</b></Link>
            </Card.Footer>
          </Card>
        </Container>
```

```
    );
};

export default Login;
```

23. Register.jsx

```
/*
@author Michal Gornicki
@Start Date 04/12/2022
*/
import React, {useState} from "react";
import {Form, Button, Container, Card, Image} from "react-bootstrap";
import {createUserWithEmailAndPassword, updateProfile} from "firebase/auth";
import {auth, db, storage} from "../firebase";
import {ref, uploadBytesResumable, getDownloadURL} from "firebase/storage";
import {doc, setDoc} from "firebase/firestore";
import {useNavigate, Link} from "react-router-dom";

const Register = () => {
    const navigate = useNavigate();

    // Form inputs
    const [firstName, setFirstName] = useState("");
    const [lastName, setLastName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [picture, setPicture] = useState(null);
    const [userRole, setUserRole] = useState("");
    const [formValid, setFormValid] = useState(false);
```

```
const [errors, setErrors] = useState({});
const [loading, setLoading] = useState(false);
const lowFirstName = firstName.toLowerCase();
const lowLastName = lastName.toLowerCase();

const buildSearchArray = (searchTerm) => {
    const searchTerms = [];
    let counter = 0;
    let term = "";
    for (let i of searchTerm) {
        term += i;
        if (counter > 0) searchTerms.push(term);
        counter += 1;
    }

    return searchTerms;
};

// Handle form submission
const handleSubmit = async (event) => {
    // prevent page refresh on form submit
    event.preventDefault();

    if (!formValid) {
        return;
    }

    try {
        // Create a user with email and password
```

```
const response = await createUserWithEmailAndPassword(
    auth,
    email,
    password
);
// get timestamp
const date = new Date().getTime();

let profileDownloadURL = null;

// if the user has uploaded a picture
if (picture) {
    // create storage reference
    const storageRef = ref(storage, `${email + picture.name + date}`);
    // upload picture
    const uploadResult = await uploadBytesResumable(storageRef, picture);
    // get download URL for the picture
    profileDownloadURL = await getDownloadURL(uploadResult.metadata.ref);
} else {
    // set default avatar URL
    profileDownloadURL = "/default-avatar.jpg";
}

try {
    // Update the user's profile
    const search = buildSearchArray(lowFirstName).concat(
        buildSearchArray(lowLastName)
    );
    await updateProfile(response.user, {
```

```
            displayName: lowFirstName + " " + lowLastName,
          });
          //create user on firestore
          await setDoc(doc(db, "users", response.user.uid), {
            uid: response.user.uid,
            displayName: response.user.displayName,
            searchArray: search,
            email: response.user.email,
            photoURL: profileDownloadURL,
            userRole: userRole,
          });

          await setDoc(doc(db, "userMessages", response.user.uid), {});

          navigate("/login");
        } catch (error) {
          // Handle error
          // console.log("Error creating user:", error);
        }
      } catch (error) {
        // Handle error
        // console.log(error);
      }
    };

    // Validate the form inputs
    const validateForm = () => {
      let valid = true;
      let errors = {};
```

```
if (!firstName || firstName.length < 2) {
    errors.firstName = (
        <Form.Text className="text-muted">
            First name must be at least 2 characters
        </Form.Text>
    );
    valid = false;
}

if (!lastName || lastName.length < 2) {
    errors.lastName = (
        <Form.Text className="text-muted">
            Last name must be at least 2 characters
        </Form.Text>
    );
    valid = false;
}

if (!email) {
    errors.email = (
        <Form.Text className="text-muted">Email address is required</Form.Text>
    );
    valid = false;
} else if (!/\S+@\S+\.\S+/.test(email)) {
    errors.email = (
        <Form.Text className="text-muted">Invalid email address</Form.Text>
    );
    valid = false;
```

```
}

if (!userRole) {
    errors.userType = (
        <Form.Text className="text-muted">User type is required</Form.Text>
    );
    valid = false;
}

// Password validation
if (!password) {
    errors.password = (
        <Form.Text className="text-muted">Password is required</Form.Text>
    );
    valid = false;
} else if (password.length < 8) {
    errors.password = (
        <Form.Text className="text-muted">
            Your password must be minimum 8 characters long
        </Form.Text>
    );
    valid = false;
} else if (!/[a-z]/.test(password)) {
    errors.password = (
        <Form.Text className="text-muted">
            Password must contain at least one lowercase letter
        </Form.Text>
    );
    valid = false;
```

```
    } else if (!/[A-Z]/.test(password)) {
      errors.password = (
        <Form.Text className="text-muted">
          Password must contain at least one uppercase letter
        </Form.Text>
      );
      valid = false;
    } else if (!/\d/.test(password)) {
      errors.password = (
        <Form.Text className="text-muted">
          Password must contain at least one number
        </Form.Text>
      );
      valid = false;
    } else if (!/[!@#\$%\^&\*]/.test(password)) {
      errors.password = (
        <Form.Text className="text-muted">
          Password must contain at least one symbol
        </Form.Text>
      );
      valid = false;
    }

    setErrors(errors);
    setFormValid(valid);
  };

  return (
    <Container className="regForm">
```

```
<Card>
  <Card.Header className="regFormCard text-center" as="h5">
    <Image src="logo.png" alt="logo" rounded/>
    Register for Creche Connect
  </Card.Header>
  <Card.Body>
    <Form onSubmit={handleSubmit}>
      <Form.Group className="mb-3" controlId="validationCustom01">
        <Form.Label>First Name</Form.Label>
        <Form.Control
          type="text"
          value={firstName}
          onChange={(e) => setFirstName(e.target.value)}
          onBlur={validateForm}
          required
        />
        {errors.firstName && (
          <div className="error-text">{errors.firstName}</div>
        )}
      </Form.Group>

      <Form.Group className="mb-3" controlId="validationCustom02">
        <Form.Label>Last Name</Form.Label>
        <Form.Control
          type="text"
          value={lastName}
          onChange={(e) => setLastName(e.target.value)}
          onBlur={validateForm}
          required
```

```
            />
            {errors.lastName && (
              <div className="error-text">{errors.lastName}</div>
            )}
          </Form.Group>

          <Form.Group className="mb-3" controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              onBlur={validateForm}
              required
            />
            {errors.email && <div className="error-text">{errors.email}</div>}
          </Form.Group>

          <Form.Group className="mb-3" controlId="validationCustom03">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              onBlur={validateForm}
              placeholder="Your password must be minimum 8 characters long,
contain letters, numbers, special characters, upper and lower cases."
              required
            />
            {errors.password && (
```

```
        <div className="error-text">{errors.password}</div>
    )}
</Form.Group>

<Form.Group className="mb-3" controlId="validationCustom04">
    <Form.Label>Add avatar</Form.Label>
    <Form.Control
        type="file"
        onChange={(e) => setPicture(e.target.files[0])}
        onBlur={validateForm}
    />
    {errors.picture && (
        <div className="error-text">{errors.picture}</div>
    )}
</Form.Group>

<Form.Group controlId="userType">
    <Form.Label>User Type</Form.Label>
    <Form.Control
        as="select"
        value={userRole}
        onBlur={validateForm}
        required
        onChange={(e) => setUserRole(e.target.value)}
    >
        <option value="">Select User Role</option>
        <option value="manager">Manager</option>
        <option value="staff">Staff</option>
        <option value="parent">Parent</option>
```

```
                    </Form.Control>
                    {errors.userType && (
                      <div className="error-text">{errors.userType}</div>
                    )}
                  </Form.Group>
                  <br></br>
                  <Container className="text-center">
                    <Button variant="primary" type="submit" size="lg" style={{margin: 5,
backgroundColor: "#f4900c",border: "#f4900c"}}>
                      Sign Up
                    </Button>
                  </Container>
                </Form>
              </Card.Body>
              <Card.Footer className="formCardFoot text-center">
                Are you already registered? <Link to="/login"><b>Login</b></Link>
              </Card.Footer>
            </Card>
          </Container>
    );
};

export default Register;
```

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
// import './index.css';
```

```
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { AuthInfo } from "./components/AuthDetails";
import { MessageInfo } from "./components/MessageDetails";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <AuthInfo>
    <MessageInfo>
      <React.StrictMode>
        <App />
      </React.StrictMode>
    </MessageInfo>
  </AuthInfo>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App.js

```
import React, {useContext} from "react";
import "./App.css";
import "bootstrap/dist/css/bootstrap.min.css";
import {BrowserRouter, Routes, Route, Navigate} from "react-router-dom";
import "./customStyle.scss";
import Login from "./pages/Login";
```

```
import Register from "./pages/Register";
import Home from "./pages/Home";
import {AuthDetails} from "./components/AuthDetails";
import Connect from "./pages/Connect";
import AddChild from "./components/AddChild";
import UpdateChild from "./components/UpdateChild";
import {updateDoc, doc} from "firebase/firestore";
import {db} from "./firebase";
import DailyReview from "./components/DailyReview";
import Archive from "./components/Archive";
import ForgotPassword from "./components/ForgotPassword";

function App() {
    const {loggedUser} = useContext(AuthDetails);

    const AuthRoute = ({children, allowedRoles}) => {
        if (loggedUser && allowedRoles.includes(loggedUser.userRole)) {
            return children;
        } else if (loggedUser) {
            return <Navigate to="/connect"/>;
        } else {
            return <Navigate to="/login"/>;
        }
    };

    const updateChild = async (updatedChild) => {
        try {
            const childRef = doc(db, "children", updatedChild.id);
            await updateDoc(childRef, updatedChild);
```

```
    } catch (error) {
        console.error("Error updating child: ", error);
    }
};
return (
    <BrowserRouter>
        <Routes>
            <Route
                path="/"
                element={
                    <AuthRoute allowedRoles={["manager", "staff"]}>
                        <Home/>
                    </AuthRoute>
                }
            />
            <Route path="/forgotpassword" element={<ForgotPassword/>}/>
            <Route path="/login" element={<Login/>}/>
            <Route path="/register" element={<Register/>}/>
            <Route
                path="/connect"
                element={
                    <AuthRoute allowedRoles={["parent", "manager", "staff"]}>
                        <Connect/>
                    </AuthRoute>
                }
            />
            <Route
                path="/addchild"
                element={
```

```
        <AuthRoute allowedRoles={["manager", "staff"]}>
          <AddChild/>
        </AuthRoute>
      }
    />
    <Route
      path="/update/:id"
      element={
        <AuthRoute allowedRoles={["manager", "staff"]}>
          <UpdateChild onUpdate={updateChild}/>
        </AuthRoute>
      }
    />
    <Route
      path="/daily-review/:id"
      element={
        <AuthRoute allowedRoles={["manager", "staff"]}>
          <DailyReview/>
        </AuthRoute>
      }
    />
    <Route
      path="/archive"
      element={
        <AuthRoute allowedRoles={["manager", "staff"]}>
          <Archive/>
        </AuthRoute>
      }
    />
```

```
          </Routes>
        </BrowserRouter>
    );
}

export default App;
```

customStyle.scss

```
/*@author Michal Gornicki
@Start date 04/12/2022  */

@mixin mobile {
  @media screen and (max-width: 480px) {
    @content;
  }
}

@mixin tablet {
  @media screen and (max-width: 768px) {
    @content;
  }
}

@mixin laptop {
  @media screen and (max-width: 1200px) {
    @content;
  }
}
```

```
@mixin desktop {
  @media screen and (min-width: 1200px) {
    @content;
  }
}

.mainmenu {

  .btn {
    width: 140px;
    background-color: #f4900c;
    border: #f4900c;
  }
}

.chat-form {
  display: flex;
  align-items: center;

  input {
    flex: 1;
    padding: 0.5rem;
    border: none;
    border-radius: 4px 0 0 4px;
    font-size: 1rem;
  }

  button {
```

```
    padding: 0.5rem 1rem;
    border: none;
    border-radius: 0 4px 4px 0;
    background-color: #4CAF50;
    color: white;
    font-size: 1rem;
    cursor: pointer;
  }
}

.forgotPass {
  margin-top: 10px;
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot a{
    color: #f4900c;
    text-decoration: none;
  }
}

.regForm {
  margin-top: 10px;
  .regFormCard{
```

```
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot a{
    color: #f4900c;
    text-decoration: none;
  }
}

.login {
  margin-top: 10px;
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot a{
    color: #f4900c;
    text-decoration: none;
  }
}
```

```
.navigation {
  .avatar {
    width: 40px;
    cursor: pointer;
    margin: 5px;
  }
}

.home {
  margin-top: 10px;

  h1 {
    color: #F4900C;
  }
}

.carousel{
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot button{
    background-color: #f4900c;
    border: #f4900c;
    margin: 5px 5px 10px 5px;
```

```
    height: 38px;
    width: 130px;
  }
}

.dailyReview{
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot button{
    background-color: #f4900c;
    border: #f4900c;
    margin: 5px;
    height: 38px;
    width: 130px;
  }
}

.addChild{
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
```

```
    background-color: black;
    color: white;
  }
  .formCardFoot button{
    background-color: #f4900c;
    border: #f4900c;
    margin: 5px;
    height: 38px;
    width: 130px;
  }
}

.updateChild{
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot button{
    background-color: #f4900c;
    border: #f4900c;
    margin: 5px;
    height: 38px;
    width: 130px;
  }
}
```

```
.archive{
  .formCard{
    background-color: black;
    color: white;
  }
  .formCardFoot{
    background-color: black;
    color: white;
  }
  .formCardFoot button{
    background-color: #f4900c;
    border: #f4900c;
    height: 38px;
    width: 130px;
  }
}


.connect {
  height: 100vh;
  margin-top: 10px;

  .wrapper {
    height: 100vh;

    .connectButton {
      margin-top: 10px;
      margin-bottom: 10px;
    }
```

```
.connectArea {
  height: calc(100% - 150px);
  display: flex;
  flex-direction: row;
  border-radius: 10px;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  overflow: hidden;

  @include mobile {
    flex-direction: column;
  }

  .sidebarCard,
  .chatCard {
    overflow-y: auto;
  }

  .sidebarCard {
    background-color: #009fff;
    padding-bottom: 20px;
    flex: 0 0 30%;
    height: 100%;

    @include mobile {
      flex: 0 0 35%; // 30% height on mobile screens
      height: 35%;
      margin-bottom: 0%;
      border-radius: 0px;
```

```
  padding-bottom: 10px;
}

.sidebar {
  background-color: #009fff;
  overflow-y: auto;

  .fixed {
    position: sticky;
    top: 0;
    z-index: 100;
    background-color: inherit;

    .search {
      .searchInput {
        height: 50px;
        border-bottom: 2px solid white;

        .inputSearch {
          background-color: transparent;
          border: none;
          cursor: pointer;
          color: white;

          &::placeholder {
            color: white;
          }
        }
      }
```

```
  }
}

.lastMessages {
  overflow-y: auto;
  margin-bottom: 10px;
}

.loggedUserMessages {
  padding: 10px;
  border-radius: 10px;
  gap: 10px;
  cursor: pointer;

  &:hover {
    background-color: #8dafce;
  }

  .avatar {
    width: 30px;
  }

  .userInfo {
    gap: 10px;
    color: #273c4d;
    cursor: pointer;
  }

  p {
```

```
        font-size: smaller;
        color: white;
      }

      span {
        font-size: large;
        font-weight: bolder;
      }
    }
  }
}

.chatCard {
  background-color: #273c4d;
  flex: 1;
  height: 100%;
  margin-bottom: 0px;
  position: relative;

  @include mobile {
    height: 65%;
    border-radius: 0px;
  }

  .chat {
    background-color: #273c4d;
    overflow-y: auto;
    padding-bottom: 50px;
```

```
  .fixed {
    position: sticky;
    top: 0;
    z-index: 100;
    background-color: inherit;

    .chatDetails {
      height: 50px;
      border-bottom: 2px solid white;
      display: flex;
      align-items: center;

      span {
        color: white;
      }
    }
  }
}

.messages {
  padding: 10px;
  overflow-y: auto;

  .message {
    display: flex;
    align-items: flex-start;
    margin-bottom: 10px;

    .messageSubject {
```

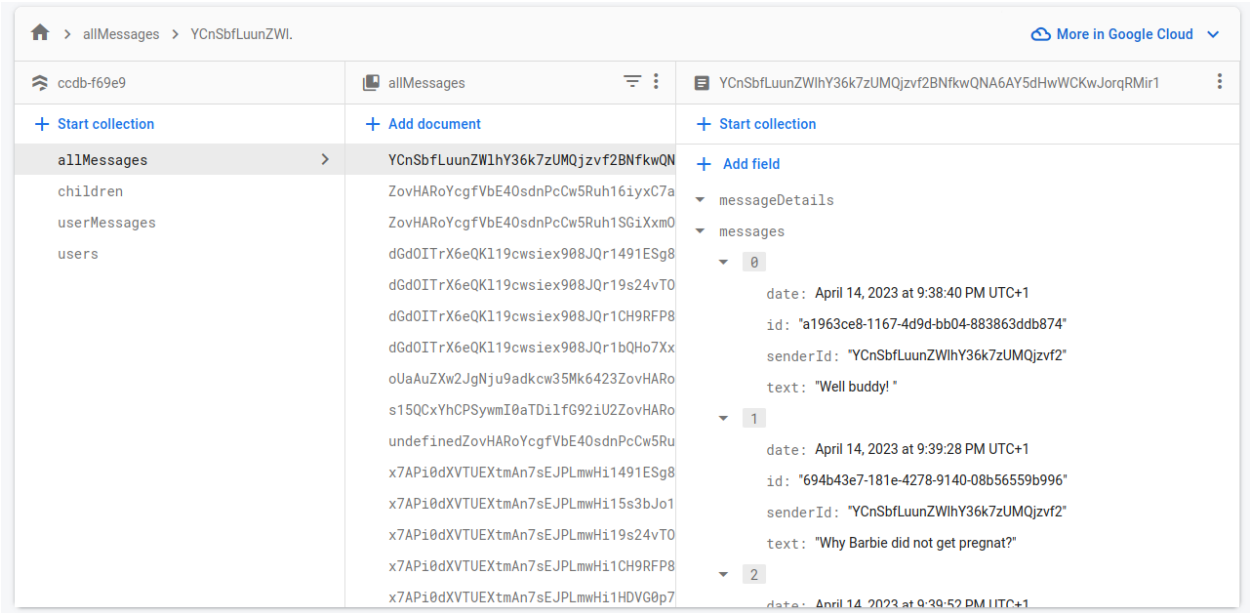
```
  // display: flex;
  // flex-direction: column;
  padding: 10px 20px;
  border-radius: 20px;

  p {
    margin: 0px;
  }

  img {
    // style for the message image, if any
    max-width: 100%;
    max-height: 300px;
    margin-top: 5px;
    margin-bottom: 5px;
    border-radius: 5px;
  }
}

&.loggedUser {
  justify-content: flex-end;

  .messageSubject {
    margin-right: 0px;
    border-radius: 20px 0px 20px 20px;
    max-width: 70%;
    background-color: #3f51b5;
    color: #fff;
  }
```

```
      .avatar {
        width: 40px;
        margin-left: 10px;
      }

    }

    &.recipient {
      justify-content: flex-start;

      .messageSubject {
        margin-left: 0px;
        border-radius: 0px 20px 20px 20px;
        max-width: 70%;
        background-color: #f0f0f0;
        color: #000;
      }

      .avatar {
        width: 40px;
        margin-right: 10px;
      }
    }
  }
}

.fixed-bottom {
  position: absolute;
```

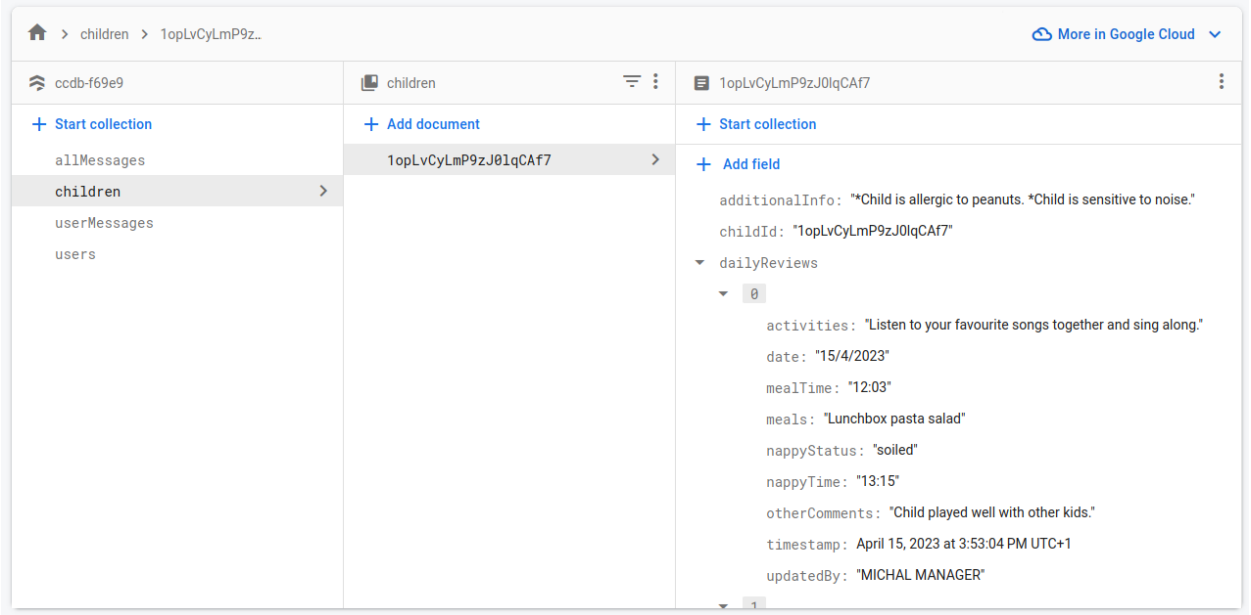
```
bottom: 0;
z-index: 100;
background-color: inherit;
padding-top: 10px;
width: 100%;

.input {
  height: 60px;
  border-top: 2px solid white;
  padding-bottom: 10px;
  display: flex;
  align-items: flex-end;

  .inputText {
    cursor: pointer;
    border-top-right-radius: 0%;
    border-bottom-right-radius: 0%;
  }

  .btn-custSend {
    padding-top: -10px;
    border-top-left-radius: 0%;
    border-bottom-left-radius: 0%;
    background-color: #f4900c;
    size: 24px;
    font-size: medium;

    &:hover {
      background-color: #b79600;
```

```
          }
        }

        .inputArea {
          background-color: white;
          font-size: medium;
          border-radius: 0%;
          border: none;
          height: 80%;
          padding-top: 6px;
          display: flex;
          align-items: center;
        }
      }
    }
  }
 }
}
```

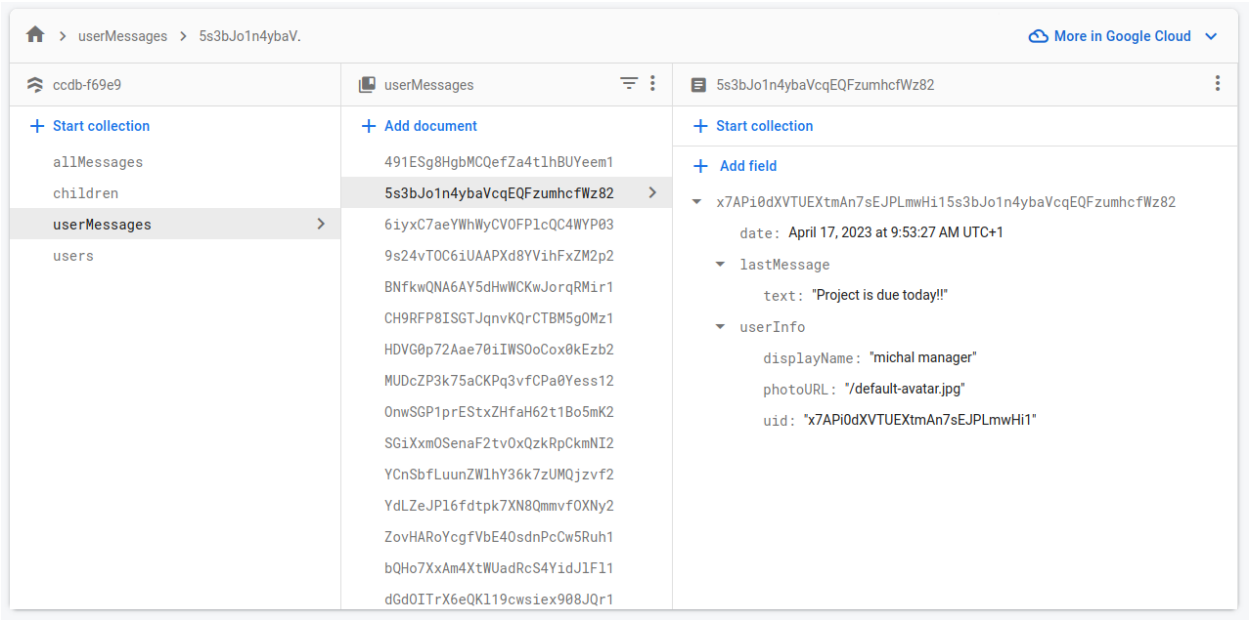
24. Database - allMessages



25. Database - children



26. Database - userMessages



27. Database - users