

D424 – Software Engineering

Task 3



Capstone Proposal Project Name: Caretaker Management Web Application

Student Name: WD

Table of Contents

<i>Application Design and Testing</i>	5
Class Design	5
UI Design	6
<i>Unit Test Plan</i>	9
Introduction	9
Purpose	9
Overview	10
Test Plan	11
Items	11
Features.....	12
Deliverables	12
Tasks	13
Needs	14
Pass/Fail Criteria.....	15
Specifications	17
Procedures	19
Results	19
<i>Hosted Web Application</i>	22
Hosted Web Application Link:	22
<i>GitLab Repository & Branch History</i>	22
GitLab Repository Link:	22
GitLab Branch history:	22

<i>User Guide for Initial Setup & Running the Application</i>	<i>23</i>
Introduction	23
Pre-requisites	23
Clone the project from Gitlab, install packages and other dependencies	25
Add domain information in Digital Ocean.....	26
Setup Domain at the Registrar	26
Add SSL with LetsEncrypt.....	27
<i>User Guide for Running the Application from User Perspective.....</i>	<i>28</i>
Introduction	28
Login	28
Dashboard page	29
Caretakers page	30
New Caretaker	31
Edit Caretaker	32
Delete Caretaker	33
Clients page	34
New Client	35
Edit Client	35
Delete Client	37
Reports page.....	38
All Caretakers	38
Specific Columns for Caretakers	39
Caretakers Over a Certain Age	40

All Clients

41

Specific Columns for Clients

42

Panopto Video Link.....

42

Application Design and Testing

Class Design

In this class diagram, the structure, and relationships of the primary components in a minimum viable product (MVP) for a Caretaker Management Web Application (CMWA) built using Node.js are represented. Since Node.js is a prototype-based language and does not have exact class type representation like traditional object-oriented languages, the diagram still serves as a useful reference for understanding the overall design and implementation of the application.

The application is designed to manage and track caretakers and clients, with the ability to generate relevant reports. The main functions in this application include Caretaker, Client, and User, which are responsible for handling their respective data and actions. Additionally, there's a Reports module for generating reports based on the caretaker and client data.

By leveraging Node.js and its prototype-based approach, this application can efficiently handle various operations and maintain a flexible, scalable architecture. The diagram showcases the attributes, methods, and relationships among these prototypes, offering a valuable guide for developers working with this Caretaker Management Web Application.



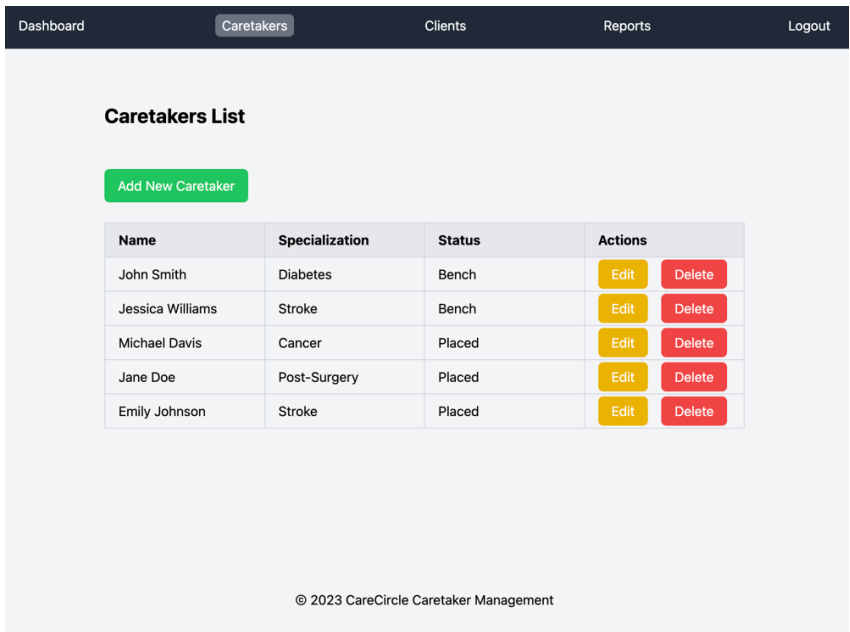
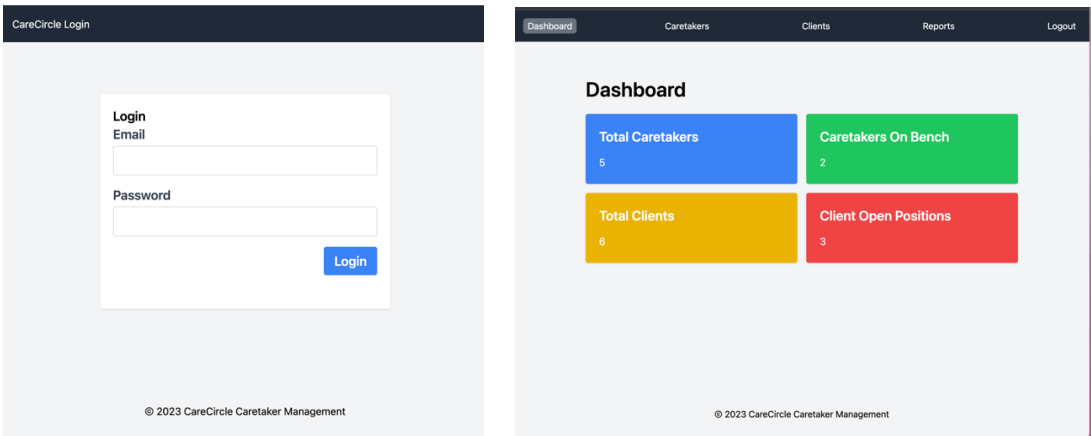
UI Design

The User Interface (UI) design of the CMWA provides a seamless and intuitive experience for users, enabling them to effectively manage and track caretakers and clients. The UI design begins with a login screen that ensures secure access to the application. Once logged in, users are taken to a dashboard that offers an overview of the most relevant.

The core features of the application are centered around managing caretakers and clients, with dedicated pages for each. Users can easily add, edit, and delete caretakers and clients using

intuitive forms and actions. The UI design is focused on providing a smooth and efficient experience, streamlining the process of managing and tracking caretakers and clients.

In addition to the core features, the application also includes a powerful reports page that enables users to generate relevant reports based on the data of caretakers and clients. This allows users to gain valuable insights and make data-driven decisions to enhance their operations.



Dashboard

Caretakers

Clients

Reports

Logout

Clients List

Add New Client

Name	City	Caretaker	Actions	
Dorothy Harris	Kolkatta	Michael Davis	Edit	Delete
Alice Green	Delhi	Jane Doe	Edit	Delete
Walter Brown	Hyderabad	Emily Johnson	Edit	Delete
Test Client	Hyderabad		Edit	Delete
Henry Miller	Kolkatta		Edit	Delete
Arthur Clark	Delhi		Edit	Delete

© 2023 CareCircle Caretaker Management

Dashboard

Caretakers

Clients

Reports

Logout

Reports Search

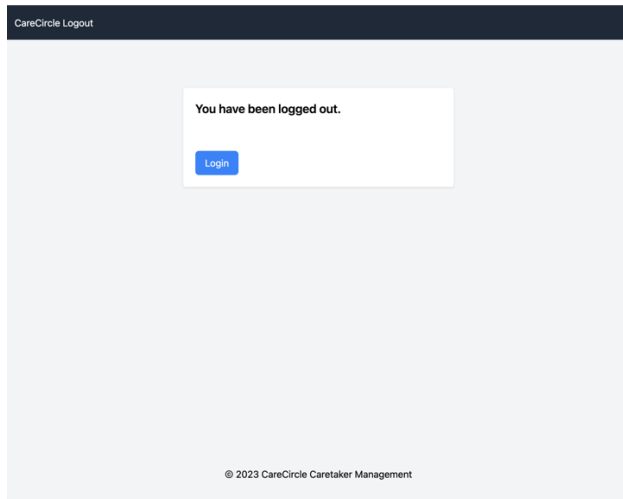
Search

--Select Type--

Search

Name	Training	Specialization	DOB	JoinDate	Status
John Smith	Level1	Diabetes	03/25/1982	02/01/2023	Bench
Jessica Williams	Level2	Stroke	09/07/1984	09/01/2019	Bench
Michael Davis	Level2	Cancer	02/14/1983	02/12/2023	Placed
Jane Doe	Level3	Post-Surgery	05/12/1980	01/01/2023	Placed
Emily Johnson	Level3	Stroke	08/30/1985	01/15/2023	Placed

© 2023 CareCircle Caretaker Management



Unit Test Plan

Introduction

Purpose

The purpose of the unit test plan is to ensure the reliability and correctness of the CMWA codebase. A systematic testing methodology was employed to isolate individual components and verify their functionality. This approach allowed for the identification and addressing of any defects, ensuring the application performs as expected and adheres to high-quality standards.

Overview

The unit test plan is an integral part of the overall development process for the CMWA. The testing method used in this plan was tailored to validate specific parts of the application, focusing on the most critical and unique aspects of the code.

Tested various functions within the application, including authentication, caretaker management, client management, and report generation. Tests covered a wide range of scenarios, such as adding, editing, and deleting caretakers and clients, as well as generating accurate reports based on the data. This thorough approach ensured that the application's features functioned correctly in isolation and together. The key functions tested included user authentication, caretaker management, client management, and report generation.

- **User Authentication:** Tested the login and logout functionality to ensure that only authorized users could access the application. The tests involved providing valid and invalid credentials, checking the application's response, and verifying that user sessions were managed correctly. When errors were encountered, such as failed authentication attempts, made necessary adjustments to the code to handle these cases gracefully and provide appropriate error messages.
- **Caretaker Management:** Tests for caretaker management covered adding, editing, and deleting caretakers in the system. Simulated form submissions with various input data to check if the application properly validated and stored the information. When errors were detected, refined the validation logic and error handling mechanisms to ensure a smooth user experience.
- **Client Management:** Like caretaker management, tests for client management covered adding, editing, and deleting clients. Verified that the application handled

client data correctly and maintained the relationship between clients and their assigned caretakers. Any errors encountered during these tests, such as incorrect data associations or validation issues, were addressed by updating the application logic and implementing proper error handling.

- **Report Generation:** Tested the report generation functionality to ensure that accurate and relevant reports could be created for both caretakers and clients.

Test Plan

Items

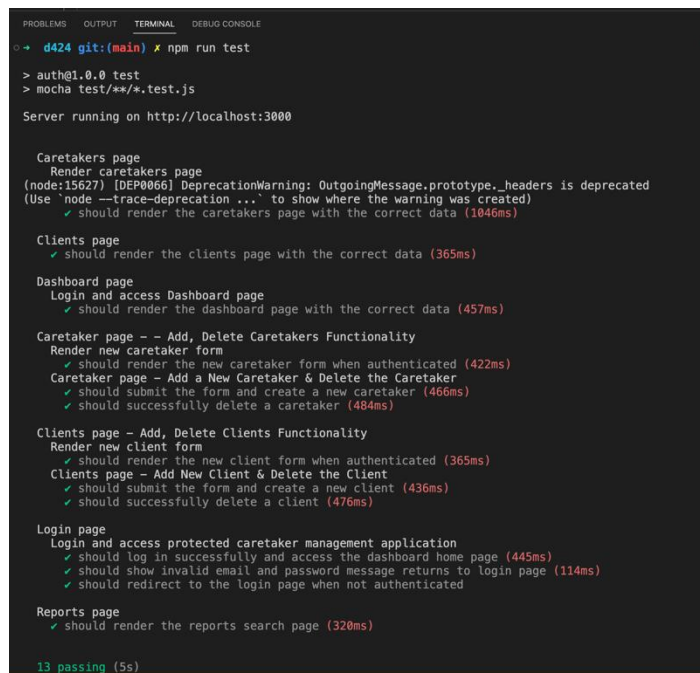
- **Development Environment:** A development environment with Node.js installed, as the application is built using Node.js.
- **Database:** Set up the application's database beforehand with prefilled data by using the provided SQL script files for data validation and manipulation during tests.
- **Test Framework:** Mocha, which is a popular testing framework for Node.js applications.
- **Assertion Library:** Chai, which is an assertion library that allows you to write more expressive and readable tests.
- **HTTP Integration Testing:** Chai-http, which is a plugin for Chai that allows you to easily perform HTTP requests for testing purposes.
- **Application Source Code:** Git clone the source code from the repository. The test scripts are already part of the codebase and will be executed using the Mocha testing framework.

Features

- User Authentication: Test user login, and logout functionality. Test access control to protected routes based on user roles.
- Caretaker Management: Test the ability to add, update, and delete caretaker records.
- Client Management: Test the ability to add, update, and delete client records.
- Reports Page: Test the generation of caretaker and client reports based on search criteria.

Deliverables

- Test Scripts: A collection of test scripts are part of the source codebase, written using the Mocha testing framework, Chai assertion library, and Chai-http for HTTP integration testing.
- Test Results: A report detailing the results of the tests, including successful tests and any failed tests with relevant error messages.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
d424 git:(main) x npm run test

> auth@1.0.0 test
> mocha test/**/*.test.js

Server running on http://localhost:3000

Caretakers page
  Render caretakers page
(node:15627) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use 'node --trace-deprecation ...' to show where the warning was created)
    ✓ should render the caretakers page with the correct data (1046ms)

Clients page
  ✓ should render the clients page with the correct data (365ms)

Dashboard page
  Login and access Dashboard page
    ✓ should render the dashboard page with the correct data (457ms)

Caretaker page -- Add, Delete Caretakers Functionality
  Render new caretaker form
    ✓ should render the new caretaker form when authenticated (422ms)
  Caretaker page -- Add a New Caretaker & Delete the Caretaker
    ✓ should submit the form and create a new caretaker (466ms)
    ✓ should successfully delete a caretaker (484ms)

Clients page -- Add, Delete Clients Functionality
  Render new client form
    ✓ should render the new client form when authenticated (365ms)
  Clients page -- Add New Client & Delete the Client
    ✓ should submit the form and create a new client (436ms)
    ✓ should successfully delete a client (476ms)

Login page
  Login and access protected caretaker management application
    ✓ should log in successfully and access the dashboard home page (445ms)
    ✓ should show invalid email and password message returns to login page (114ms)
    ✓ should redirect to the login page when not authenticated

Reports page
  ✓ should render the reports search page (320ms)

13 passing (5s)
```

Tasks

- Set up the development environment with the required tools, including Node.js, Mocha, Chai, and Chai-http.
- Ensure test scripts for each feature outlined in the Test Plan are in place, such as user authentication, caretaker management, client management, and the reports page. Since this is an MVP, the test scripts are already part of the codebase and kept minimal.
- Prepare test data to be used during testing by using the prefilled data from the SQL file provided in the source codebase. This includes user credentials, sample caretaker information, and sample client details.
- Execute the test scripts using the Mocha testing framework and monitor the test results.
- Analyze the test results and identify any failed tests or unexpected behavior. The completed tests results would be as below:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
o → d424 git:(main) x npm run test

> auth@1.0.0 test
> mocha test/**/*.test.js

Server running on http://localhost:3000

Caretakers page
  Render caretakers page
(node:15627) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
    ✓ should render the caretakers page with the correct data (1046ms)

Clients page
  ✓ should render the clients page with the correct data (365ms)

Dashboard page
  Login and access Dashboard page
    ✓ should render the dashboard page with the correct data (457ms)

Caretaker page -- Add, Delete Caretakers Functionality
  Render new caretaker form
    ✓ should render the new caretaker form when authenticated (422ms)
  Caretaker page -- Add a New Caretaker & Delete the Caretaker
    ✓ should submit the form and create a new caretaker (466ms)
    ✓ should successfully delete a caretaker (484ms)

Clients page -- Add, Delete Clients Functionality
  Render new client form
    ✓ should render the new client form when authenticated (365ms)
  Clients page -- Add New Client & Delete the Client
    ✓ should submit the form and create a new client (436ms)
    ✓ should successfully delete a client (476ms)

Login page
  Login and access protected caretaker management application
    ✓ should log in successfully and access the dashboard home page (445ms)
    ✓ should show invalid email and password message returns to login page (114ms)
    ✓ should redirect to the login page when not authenticated

Reports page
  ✓ should render the reports search page (320ms)

13 passing (5s)
```

Needs

- **Software Requirements:** The application's dependencies, such as Node.js, Express, and any other required libraries, must be installed with the appropriate versions. This ensures compatibility and prevents issues related to version mismatches.
- **Testing Tools and Libraries:** For this project, Mocha as the testing framework, Chai for assertions, and Chai-HTTP for testing HTTP requests are used. These tools need to be installed and configured correctly to ensure smooth test execution.

- Access to Source Code and Test Scripts: Access to the application's source code and test scripts to review and update them as needed during the testing process. This may involve setting up a version control system, such as Git, to manage code changes and keep track of test-related modifications.
- Test data to be used during testing by using the prefilled data from the SQL file provided in the codebase. This includes user credentials, sample caretaker information, and sample client details.

Pass/Fail Criteria

- User Authentication
 - Pass: User is successfully authenticated with valid email and password.
 - Fail: User is not authenticated due to incorrect email and password.
- Dashboard:
 - Pass: Dashboard displays correct information.
 - Fail: Dashboard does not display the correct information or is not accessible.
- Caretakers:
 - Pass: Caretakers list is displayed correctly, and add, edit, and delete functionalities work as intended.
 - Fail: Caretakers list is not displayed.
- Clients:
 - Pass: All clients list is displayed correctly, and add, edit, and delete functionalities work as intended.
 - Fail: All clients list is not displayed.

- Reports Page Tests:
 - Pass: Reports are generated and displayed correctly.
 - Fail: Reports are not generated.

If a test failed during the testing process, the following remediation strategies and documentation requirements were applied to address the issue:

- Identify the root cause: Investigate the failed test to determine the underlying cause of the issue. Examine the error messages, logs, or any other relevant information to understand the nature of the problem.
- Document the issue: Create a bug report or issue a ticket to document the problem. Include details such as the specific test that failed, the error message, reproduction steps, and any other relevant information.

Here are some examples of test code screenshots taken from the CMWA source codebase.

```
login and access protected caretaker management application } callback } it('should log in successfully and access the dashboard home page', () => {
  const chai = require('chai')
  const chaiHttp = require('chai-http')
  const app = require('../app')
  const { expect } = chai

  chai.use(chaiHttp)

  describe('Login page', () => {
    describe('Login and access protected caretaker management application', () => {
      it('should log in successfully and access the dashboard home page', (done) => {
        const agent = chai.request.agent(app)
        agent
          .post('/login')
          .send({ email: 'a@a.a', password: 'aa' })
          .then((res) => {
            expect(res).to.have.status(200)
            return agent.get('/').then((res) => {
              expect(res).to.have.status(200)
              expect(res.text).to.include('Dashboard')
              done()
            })
          })
          .catch((err) => {
            done(err)
          })
      })

      it('should show invalid email and password message returns to login page', (done) => {
        const agent = chai.request.agent(app)
        agent
          .post('/login')
          .send({ email: 'a@a.a', password: 'a' })
          .end((err, res) => {
            expect(err).to.be.null
            expect(res).to.have.status(401)
            expect(res.text).to.include('Invalid email or password')
            done()
          })
      })
    })
  })
})
```

```
describe('Caretaker page - Add a New Caretaker & Delete the Caretaker', () => {
  it('should submit the form and create a new caretaker', (done) => {
    const agent = chai.request.agent(app)

    agent
      .post('/login')
      .send({ email: 'a@a.a', password: 'aa' })
      .then((res) => {
        expect(res).to.have.status(200)

        return agent
          .post('/caretakers')
          .send({
            name: 'Test Caretaker',
            dob: '1980-01-01',
            gender: 'Female',
            training: 'Level1',
            skills: 'Test skill',
            joindate: '2023-03-16',
            specialization: 'Bed-ridden',
            updatedbyid: '0878be02-abfa-4987-8b38-d8e79c9fb415',
          })
          .then((res) => {
            expect(res).to.have.status(200)

            return agent.get('/caretakers').then((res) => {
              expect(res).to.have.status(200)
              expect(res.text).to.include('Test Caretaker')
              expect(res.text).to.include('Bed-ridden')
              done()
            })
          })
      })
      .catch((err) => {
        done(err)
      })
  })
})
```

```
it('should successfully delete a caretaker', async () => {
  const agent = chai.request.agent(app)
  await agent.post('/login').send({ email: 'a@a.a', password: 'aa' })

  // Get the newly added caretaker ID from the database
  const caretakers = await db.query(
    'SELECT id FROM caretakers WHERE name = $1',
    ['Test Caretaker']
  )
  const caretakerId = caretakers[0].id
  await agent.post(`/caretakers/${caretakerId}/delete`)
  const res = await agent.get('/caretakers')
  expect(res).to.have.status(200)
  expect(res.text).not.to.include('Test Caretaker')
})
```

Procedures

- Test case preparation: Identified the critical functionalities and features of the application that require testing. Wrote test cases to cover various scenarios, Defined the expected outcomes and pass/fail criteria for each test case.
- Test environment setup: Testing environment setup, including any necessary dependencies, libraries, and frameworks. Verified that the Node.js application and test tools (e.g., Mocha, Chai, Chai-HTTP) are properly installed and configured.
- Test execution: Ran the test suite using the test runner (e.g., Mocha) and record the test results. Analyze the test results and identify any failed tests. Iteratively refine the test cases and application code as needed to address any identified issues or failures.
- Test results review and documentation: Documented the test results, including pass/fail outcomes, identified issues, and any modifications made to the application code or test cases. Update the test plan and test case documentation to reflect the final test outcomes.

Results

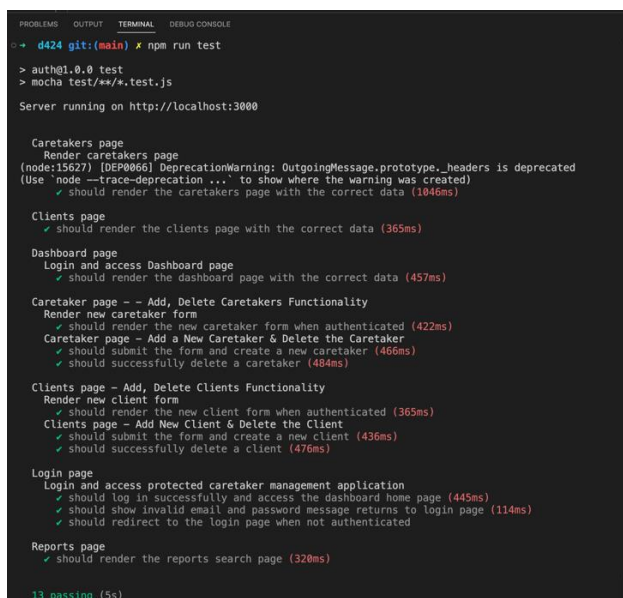
The test results for the CMWA were obtained after executing the test scripts using the Mocha testing framework. The following are examples of the testing results:

- User Authentication: The test passed when the user was able to successfully log in using valid email and password combinations, and the server returned a status code of 200. Login with invalid credentials: The test passed when the user attempted to log in with invalid email or password combinations, and the server returned an appropriate error message along with a status code of 401 (Unauthorized).

- Caretaker Management: The test passed when a new caretaker was successfully added to the system, and their information was stored in the database. The server returned a status code of 200, and the added caretaker's details were present when querying the 'caretakers' endpoint. The test passed when an existing caretaker was successfully removed from the system, and their information was deleted from the database. The server returned a status code of 200, and the deleted caretaker's details were no longer present when querying the 'caretakers' endpoint.
- Client Management: The test passed when a new client was successfully added to the system with a valid age, and their information was stored in the database. The server returned a status code of 200, and the added client's details were present when querying the 'clients' endpoint. Add a new client with an invalid age: The test passed when attempting to add a new client with an invalid age resulted in an AssertionError, indicating that the validation rules for age were working as intended.

These test results helped to ensure the reliability and correctness of the CMWA codebase.

Results Screenshot - all tests passed.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
d424 git:(main) x npm run test
> auth@1.0.0 test
> mocha test/**/*.test.js

Server running on http://localhost:3000

Caretakers page
  Render caretakers page
    (node:15627) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
    (Use 'node --trace-deprecation ...' to show where the warning was created)
    ✓ should render the caretakers page with the correct data (1046ms)

Clients page
  ✓ should render the clients page with the correct data (365ms)

Dashboard page
  Login and access Dashboard page
    ✓ should render the dashboard page with the correct data (457ms)

Caretaker page - Add, Delete Caretakers Functionality
  Render new caretaker form
    ✓ should render the new caretaker form when authenticated (422ms)
  Caretaker page - Add a New Caretaker & Delete the Caretaker
    ✓ should submit the form and create a new caretaker (466ms)
    ✓ should successfully delete a caretaker (404ms)

Clients page - Add, Delete Clients Functionality
  Render new client form
    ✓ should render the new client form when authenticated (365ms)
  Clients page - Add New Client & Delete the Client
    ✓ should submit the form and create a new client (436ms)
    ✓ should successfully delete a client (476ms)

Login page
  Login and access protected caretaker management application
    ✓ should log in successfully and access the dashboard home page (445ms)
    ✓ should show invalid email and password message returns to login page (114ms)
    ✓ should redirect to the login page when not authenticated

Reports page
  ✓ should render the reports search page (320ms)

13 passing (5s)
```

Results Screenshot – for invalid age input and how it fails for that. Based on these kinds of assertion the codebase was fixed.

```
31 //
32
33 describe('Clients page - Add New Client & Delete the Client', () => {
34   it.only('should submit the form and create a new client', (done) => {
35     const agent = chai.request.agent(app)
36
37     agent
38       .post('/login')
39       .send({ email: 'a@a.a', password: 'aa' })
40       .then((res) => {
41         expect(res).to.have.status(200)
42         return agent
43           .post('/clients')
44           .send({
45             name: 'Test Client',
46             city: 'Hyderabad',
47             age: '80a',
48             gender: 'Female',
49             need: 'Bed-ridden',
50             updatedbyid: '0878be02-abfa-4987-8b38-d8e79c9fb415',
51           })
52           .then((res) => {
53             expect(res).to.have.status(200)
54             return agent.get('/clients').then((res) => {
55               expect(res).to.have.status(200)
56               expect(res.text).to.include('Test Client')
57               expect(res.text).to.include('Hyderabad')
58               done()
59             })
60           })
61       })
62       .catch((err) => {
63         done(err)
64       })
65   })
66 })
```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE

```
1) Clients page - Add, Delete Clients Functionality
   Clients page - Add New Client & Delete the Client
     should submit the form and create a new client:

AssertionError: expected { Object (_events, _eventsCount, ...) } to have status code 200 but got 400
+ expected - actual

-400
+200
```

Hosted Web Application

Hosted Web Application Link:

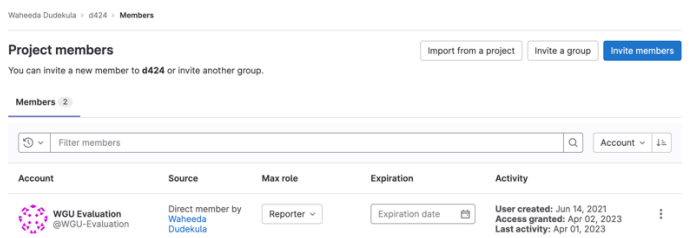
<https://www.gitagandhi.org/>

GitLab Repository & Branch History

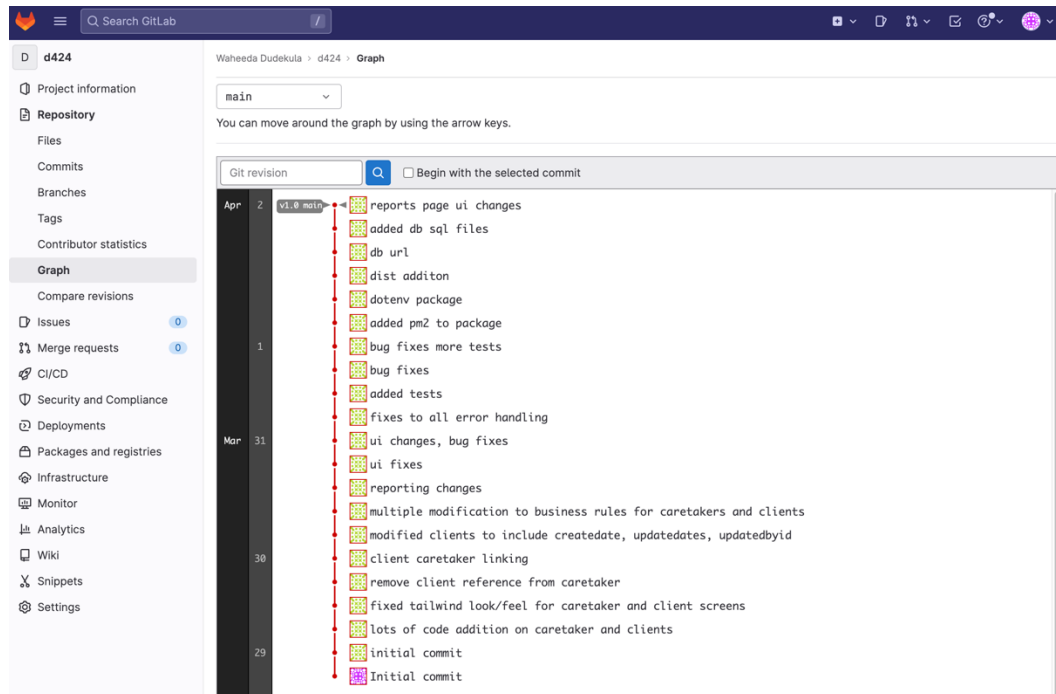
GitLab Repository Link:

<https://gitlab.com/wdudeku/d424.git>

Added WGU-Evaluation as a member



GitLab Branch history:



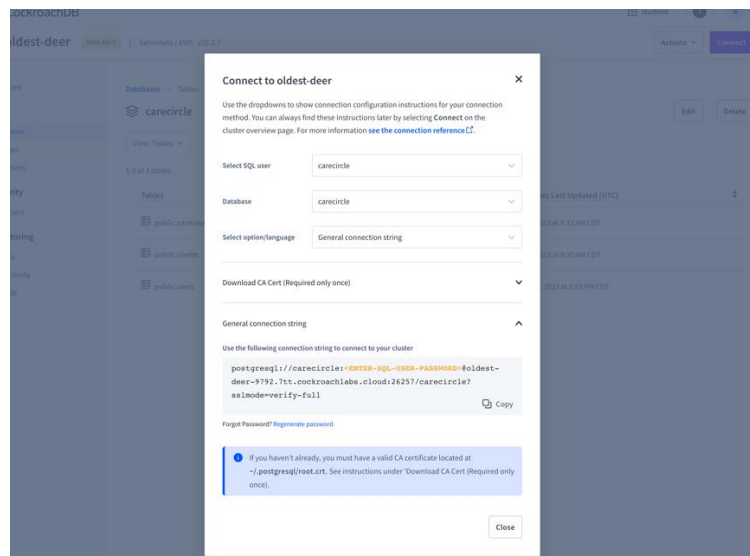
User Guide for Initial Setup & Running the Application

Introduction

This User Guide presents a thorough overview of the initial setup and steps required to get the application up and running.

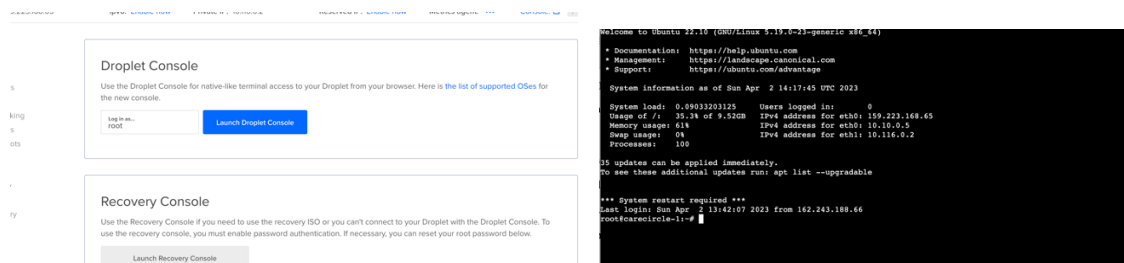
Pre-requisites

1. Register the website domain at a website registrar such as <https://www.namecheap.com/>.
2. An account setup on Cockroach DB (<https://cockroachlabs.cloud/clusters>) and setup the database.
 - a. Copy the database connection URL (you can access the information using the “Connect” on the CockroachDB portal)



- b. Setup CockroachDB cli on your local machine using this link - <https://www.cockroachlabs.com/docs/cockroachcloud/quickstart>)
- c. Using the CockroachDB cli create the tables using the schema_db.sql from the source codebase. Copy and paste the 3 tables create script in the cli window.

- d. Using the CockroachDB cli create the tables using the reset_db.sql from the source codebase. Copy and insert statements scripts into the cli window.
3. DigitalOcean cloud platform account setup (<https://www.digitalocean.com/>)
 - a. Create a new droplet for “carecircle”
 - i. Choose Ubuntu (Default)
 - ii. Choose the plan of your choice (\$5/mo or \$10/mo)
 - iii. Choose default region
 - iv. Select Authentication - SSH keys (for mac/windows or your version of the local environment)
 - v. Click Create Droplet
 - b. Launch “Droplet” console window and run the following commands. This gets the latest updates, installs nodejs and then installs npm.
 - i. `sudo apt update`
 - ii. `sudo apt install nodejs`
 - iii. `node --version`
 - iv. `sudo apt install npm`
 - v. `npm --version`



Clone the project from Gitlab, install packages and other dependencies

1. Continue these steps into the DigitalOcean's "Droplet" console window.
 - a. `git clone https://gitlab.com/wdudeku/d424.git`
 - b. `npm install`
 - c. `touch .env`
 - d. edit the .env file and add the database URL (from prerequisite step 2 above) into the .env file and save the file. For example:

`DATABASE_URL_CC=postgresql://carecircle:ZE5RTo-rQ790hD-cS9CT6Q@oldest-deer-9792.7tt.cockroachlabs.cloud:26257/carecircle?sslmode=verify-full`
2. Setup PM2 process manager to keep the application running
 - a. `npm i pm2 -g`
3. Setup ufw firewall
 - a. `sudo ufw enable`
 - b. `sudo ufw status`
 - c. `sudo ufw allow ssh (Port 22)`
 - d. `sudo ufw allow http (Port 80)`
 - e. `sudo ufw allow https (Port 443)`
4. Install NGINX and configure (necessary for https)
 - a. `sudo apt install nginx`
 - b. `sudo nano /etc/nginx/sites-available/default`
 - c. Add the following to the location part of the server block

`server_name yourdomain.com www.yourdomain.com;`

`location / {`

```
proxy_pass http://localhost:5000; #whatever port your app runs on

proxy_http_version 1.1;

proxy_set_header Upgrade $http_upgrade;

proxy_set_header Connection 'upgrade';

proxy_set_header Host $host;

proxy_cache_bypass $http_upgrade;

}
```

- d. Run the following commands and you should now be able to visit your IP with no port (port 80) and see the application.

- i. `sudo nginx -t`
- ii. `sudo service nginx restart`

Add domain information in Digital Ocean

1. In Digital Ocean, go to networking and add a domain.
2. Add an A record for @ and for www to your droplet.

Setup Domain at the Registrar

1. Add the following 3 custom nameservers in your registrar portal (such as namecheap.com):
 - a. `ns1.digitalocean.com`
 - b. `ns2.digitalocean.com`
 - c. `ns3.digitalocean.com`

Add SSL with LetsEncrypt

1. Continue these steps into the “Droplet” console window:
 - a. `sudo add-apt-repository ppa:certbot/certbot`
 - b. `sudo apt-get update`
 - c. `sudo apt-get install python-certbot-nginx`
 - d. `sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com`
2. Start the application using this command
 - a. `npm run server`
3. Open a web browser to see the application given the domain url
(<https://www.yourdomain.com>)

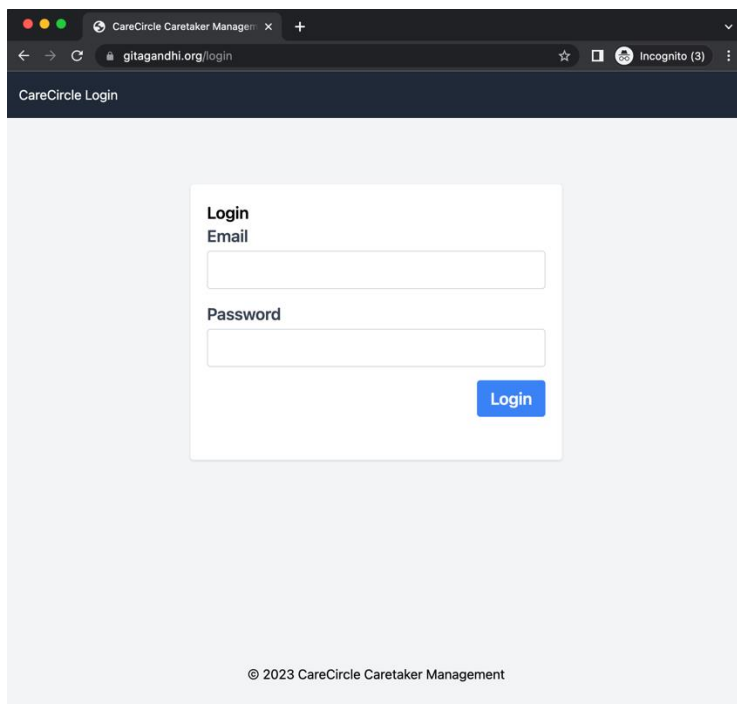
User Guide for Running the Application from User Perspective

Introduction

This user guide is designed to guide you through the login process and offer a comprehensive overview of all available functions within the application.

Login

There are two distinct methods for accessing the application, each tailored to a specific user type - staff users and admin users. The respective login credentials for these user roles are as follows:

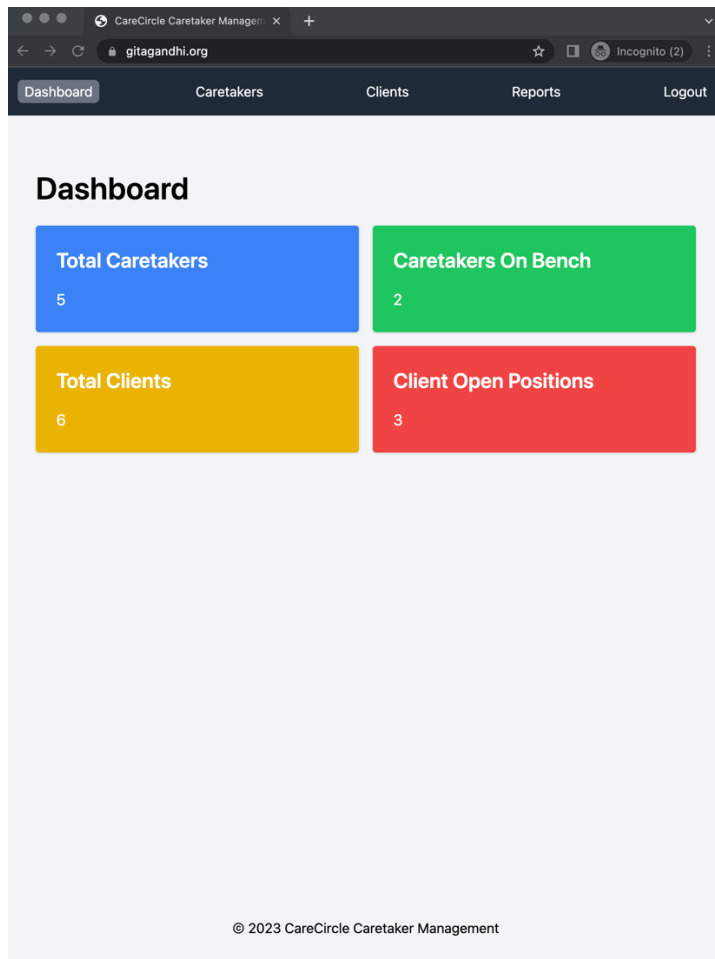


ccadmin@cc.com / ccadmin1234 (this user will have access to Clients page)

ccstaff@cc.com / ccstaff1234 (this user will **not** have access to Clients page)

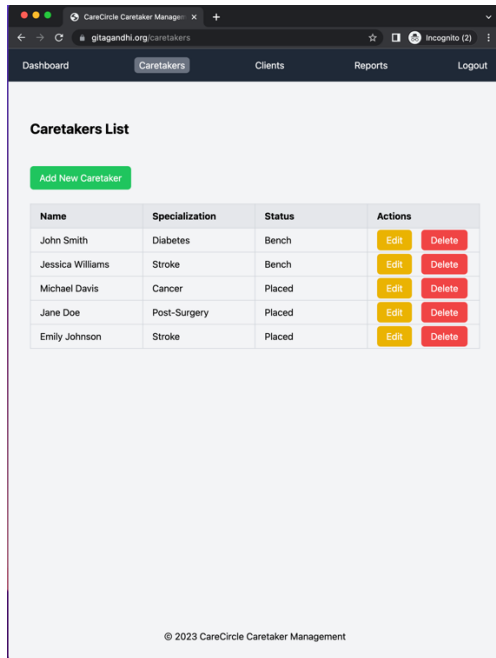
Dashboard page

Once logged in the application takes you to the dashboard page.



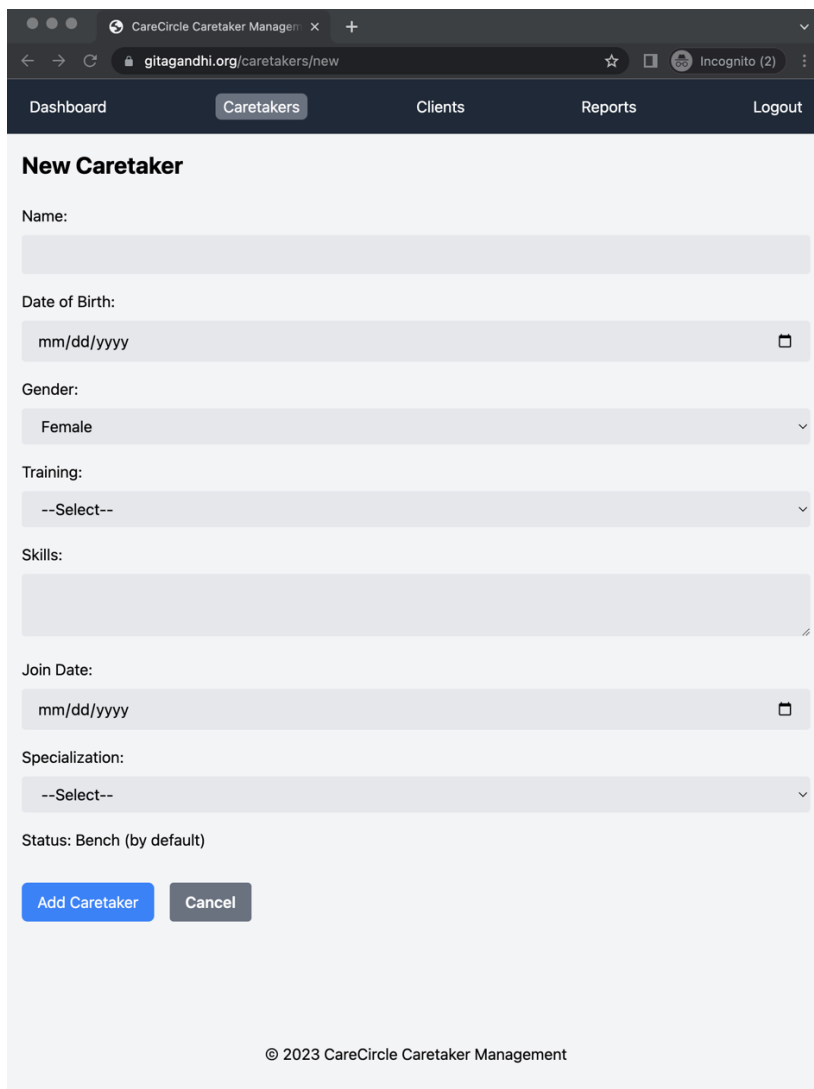
Caretakers page

To manage caretakers, such as adding, editing, or deleting, simply navigate to all caretakers page by selecting the "Caretakers" option from the menu.



New Caretaker

Select the "Add New Caretaker" button to begin the process of adding a new caretaker. Please ensure all fields are completed on this page. Upon clicking the "Add Caretaker" button, the application will redirect you to the "Caretakers page," where you should now see the newly added caretaker in the list.



The screenshot shows a web browser window with the URL `gitagandhi.org/caretakers/new`. The application has a dark blue navigation bar with links for **Dashboard**, **Caretakers** (active), **Clients**, **Reports**, and **Logout**. The main content area is titled **New Caretaker** and contains the following form fields:

- Name:** A text input field.
- Date of Birth:** A date picker showing `mm/dd/yyyy`.
- Gender:** A dropdown menu with `Female` selected.
- Training:** A dropdown menu with `--Select--` selected.
- Skills:** A text area for entering skills.
- Join Date:** A date picker showing `mm/dd/yyyy`.
- Specialization:** A dropdown menu with `--Select--` selected.
- Status:** A label indicating `Bench (by default)`.

At the bottom of the form are two buttons: **Add Caretaker** (blue) and **Cancel** (grey). The footer of the page reads `© 2023 CareCircle Caretaker Management`.

Edit Caretaker

To edit an existing caretaker, navigate to the "Caretakers page" and locate the caretaker you wish to update. Click the "Edit" button corresponding to the desired caretaker. Ensure all fields are completed on the editing page. Once you click the "Save Changes" button, the application will redirect you to the "Caretakers page," where you should now see the updated information for the selected caretaker.

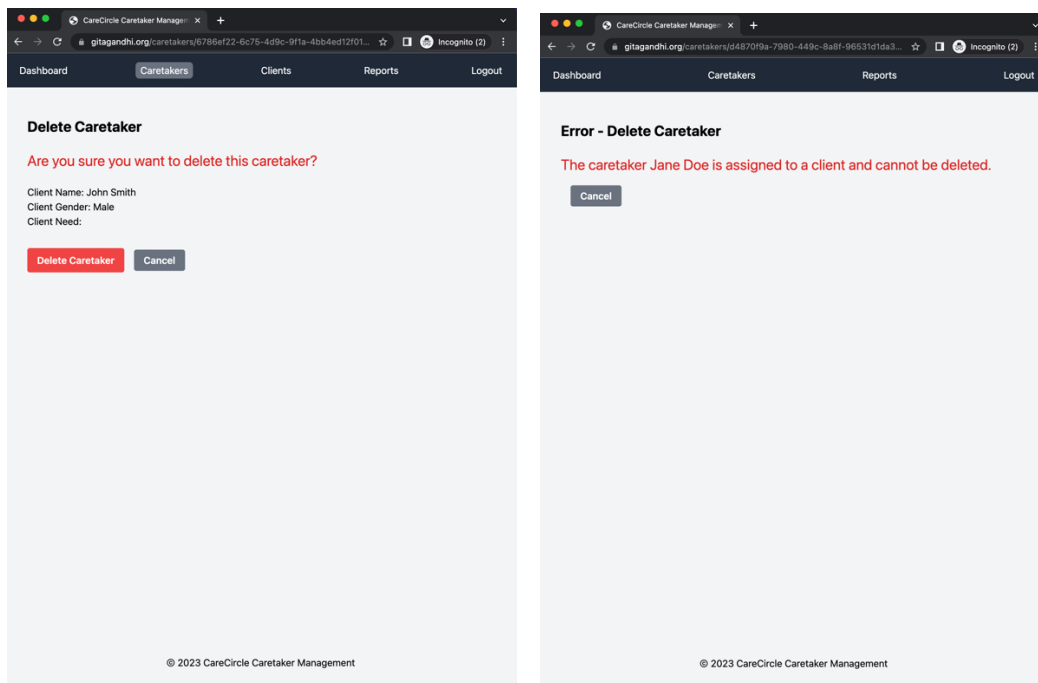
The screenshot shows a web browser window with the title "CareCircle Caretaker Management". The address bar shows the URL "gitagandhi.org/caretakers/6786ef22-6c75-4d9c-9f1a-4bb4ed12f01...". The browser is in Incognito mode. The application has a dark blue navigation bar with links: Dashboard, Caretakers (active), Clients, Reports, and Logout. The main content area is titled "Edit Caretaker" and contains the following form fields:

- Name: John Smith
- Date of Birth: 03/25/1982
- Gender: Male
- Training: Level1
- Skills: CPR, First Aid
- Join Date: 02/01/2023
- Specialization: Diabetes
- Status: Bench

At the bottom of the form are two buttons: "Update Caretaker" (blue) and "Cancel" (grey). The footer of the page reads "© 2023 CareCircle Caretaker Management".

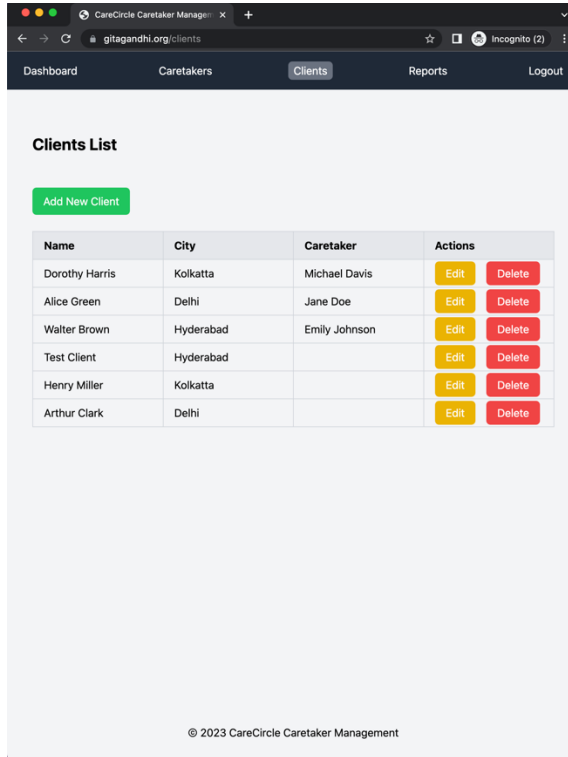
Delete Caretaker

To delete a caretaker, go to the "Caretakers page" and find the caretaker you want to remove. Click the "Delete" button corresponding to the caretaker you wish to delete. Note that caretakers with the status "Placed" cannot be deleted. For caretakers with other statuses such as Bench, this action will direct you to a confirmation page where you'll be asked to confirm your decision to delete the caretaker. Once you confirm the deletion by clicking the "Delete Caretaker" button, the application will remove the caretaker from the list and return you to the updated "Caretakers page," where the deleted caretaker should no longer be visible.



Clients page

To add, edit, and delete clients, go to the "Clients" page by clicking on the "Clients" menu.

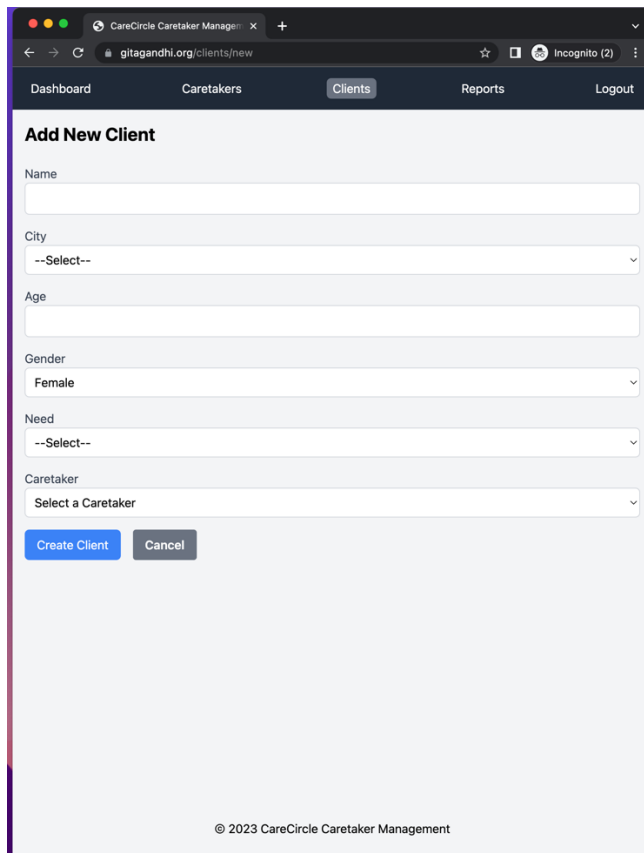


The screenshot shows a web browser window with the URL `gitagandhi.org/clients`. The application has a dark navigation bar with links for Dashboard, Caretakers, Clients (active), Reports, and Logout. The main content area is titled "Clients List" and features a green "Add New Client" button. Below the button is a table with columns for Name, City, Caretaker, and Actions. The table contains seven rows of client data, each with "Edit" and "Delete" buttons in the Actions column. The footer of the page displays the copyright notice "© 2023 CareCircle Caretaker Management".

Name	City	Caretaker	Actions
Dorothy Harris	Kolkatta	Michael Davis	Edit Delete
Alice Green	Delhi	Jane Doe	Edit Delete
Walter Brown	Hyderabad	Emily Johnson	Edit Delete
Test Client	Hyderabad		Edit Delete
Henry Miller	Kolkatta		Edit Delete
Arthur Clark	Delhi		Edit Delete

New Client

To add a new client, click the "Add New Client" button. All fields are required except the Caretaker drop-down. Once you click the "Add Client" button, the application takes you to all "Clients page," and you should see your newly added client.



The screenshot shows a web browser window with the URL `gitagandhi.org/clients/new`. The application has a dark navigation bar with links for Dashboard, Caretakers, Clients (active), Reports, and Logout. The main content area is titled "Add New Client" and contains the following fields:

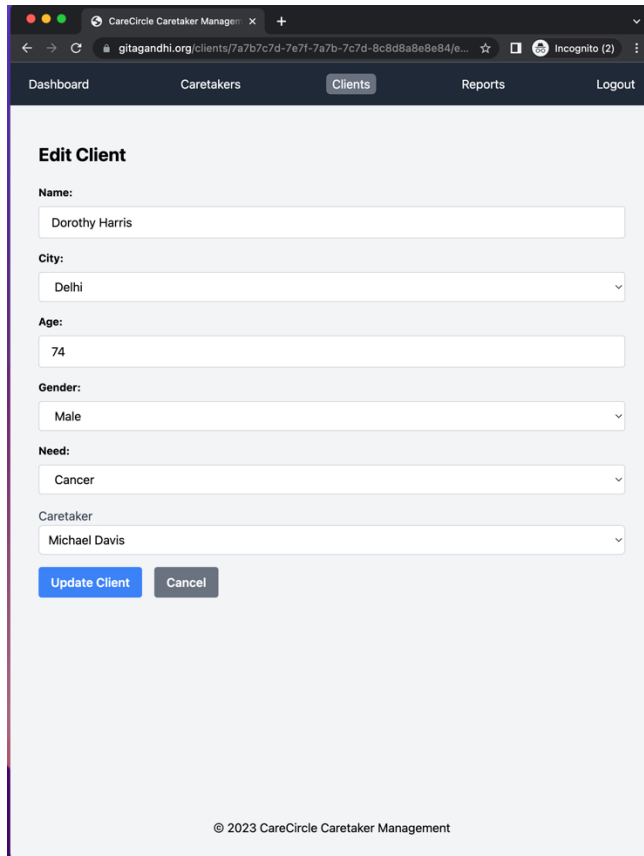
- Name: A text input field.
- City: A dropdown menu with "--Select--" as the current selection.
- Age: A text input field.
- Gender: A dropdown menu with "Female" as the current selection.
- Need: A dropdown menu with "--Select--" as the current selection.
- Caretaker: A dropdown menu with "Select a Caretaker" as the current selection.

At the bottom of the form are two buttons: "Create Client" (blue) and "Cancel" (grey). The footer of the page reads "© 2023 CareCircle Caretaker Management".

Edit Client

To edit an existing client, go to the "Clients page" and find the client you want to modify. Click the "Edit" button corresponding to the client you wish to edit. This is where you can assign a caretaker to a client, as well as make any other desired changes to the client's information. After making the necessary adjustments, click the "Save Changes" button. The application will then

update the client's information and return you to the "Clients page," where you can see the updated information.



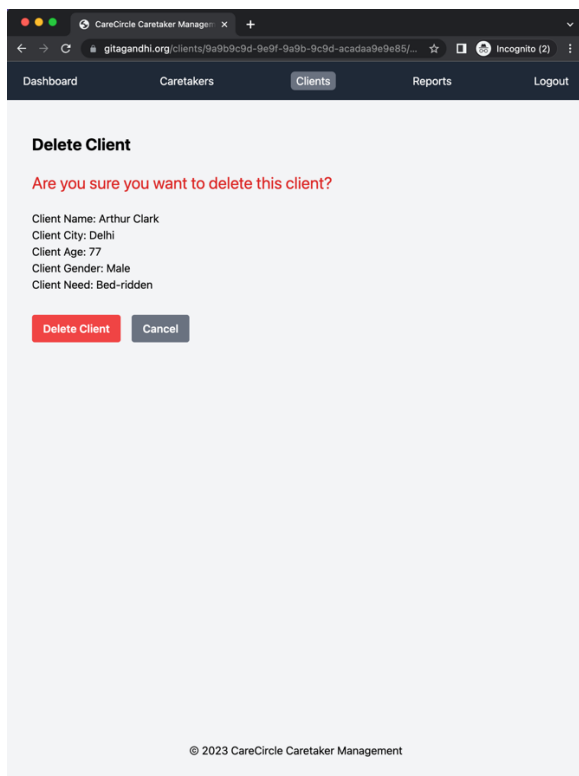
The screenshot shows a web browser window with the title "CareCircle Caretaker Management". The address bar shows a URL from gitagandhi.org. The navigation bar includes links for Dashboard, Caretakers, Clients (active), Reports, and Logout. The main content area is titled "Edit Client" and contains the following form fields:

- Name:** Text input field containing "Dorothy Harris".
- City:** Dropdown menu showing "Delhi".
- Age:** Text input field containing "74".
- Gender:** Dropdown menu showing "Male".
- Need:** Dropdown menu showing "Cancer".
- Caretaker:** Dropdown menu showing "Michael Davis".

At the bottom of the form are two buttons: "Update Client" (blue) and "Cancel" (grey). The footer of the page reads "© 2023 CareCircle Caretaker Management".

Delete Client

To delete a client, go to the "Clients page" and find the client you want to remove. Click the "Delete" button corresponding to the client you wish to delete. This action will direct you to a confirmation page where you'll be asked to confirm your decision to delete the client. Once you confirm the deletion by clicking the "Delete Client" button, the application will remove the client from the list and return you to the updated "Clients page," where the deleted client should no longer be visible.

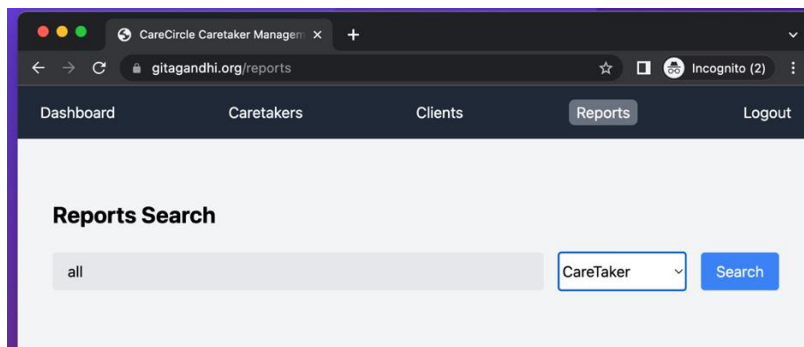


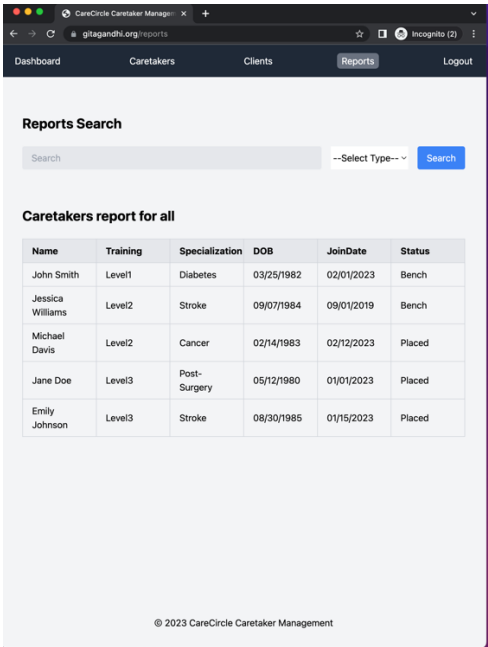
Reports page

On the "Reports" page, you can generate various reports that offer insights into both caretakers and clients. By clicking on the "Reports" menu, you will be directed to the "Reports" page, where the following report options are available:

All Caretakers

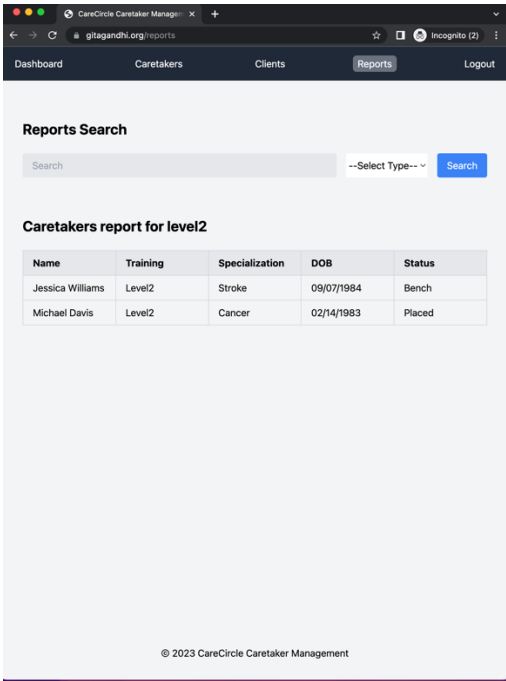
This report displays comprehensive information about all caretakers in the system. To generate this report, type "all" in the search field, select "caretaker" from the drop-down menu, and then click the "Search" button. The application will display a comprehensive list of caretakers and their associated information.





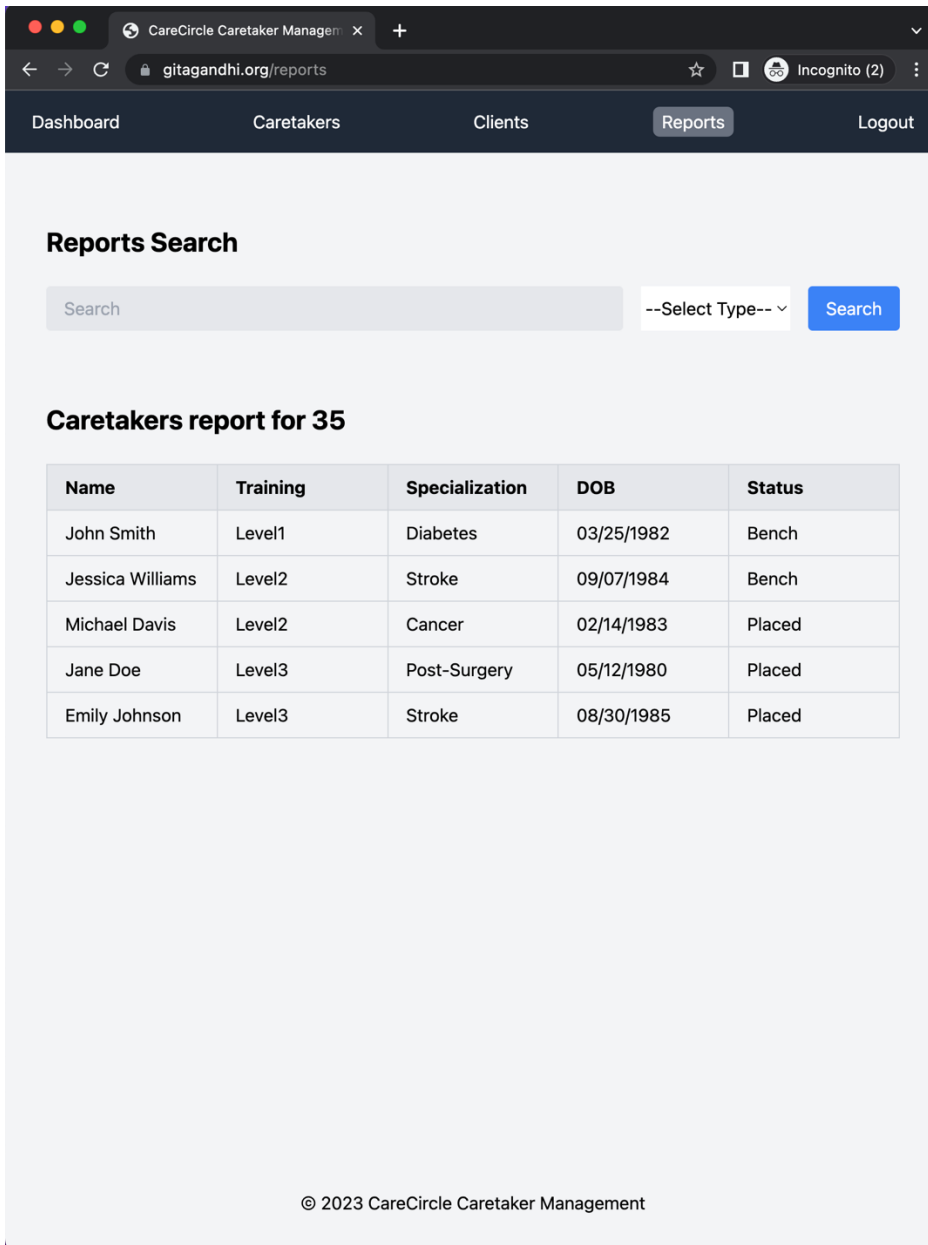
Specific Columns for Caretakers

This report allows you to select specific columns related to caretakers, displaying only the chosen information.



Caretakers Over a Certain Age

This report allows you to filter caretakers based on their age. For example, by entering the age say 35, the report will return all caretakers who are over 35 years old.



The screenshot shows a web browser window with the URL `gitagandhi.org/reports`. The application has a dark navigation bar with links for Dashboard, Caretakers, Clients, Reports (active), and Logout. Below the navigation bar, there is a 'Reports Search' section with a search input field, a dropdown menu set to '--Select Type--', and a blue 'Search' button. The main content area displays a table titled 'Caretakers report for 35'.

Name	Training	Specialization	DOB	Status
John Smith	Level1	Diabetes	03/25/1982	Bench
Jessica Williams	Level2	Stroke	09/07/1984	Bench
Michael Davis	Level2	Cancer	02/14/1983	Placed
Jane Doe	Level3	Post-Surgery	05/12/1980	Placed
Emily Johnson	Level3	Stroke	08/30/1985	Placed

© 2023 CareCircle Caretaker Management

All Clients

This report shows comprehensive information about all clients in the system.

CareCircle Caretaker Management

gitagandhi.org/reports

Incognito (2)

DashboardCaretakersClientsReportsLogout

Reports Search

Search

--Select Type--

Search

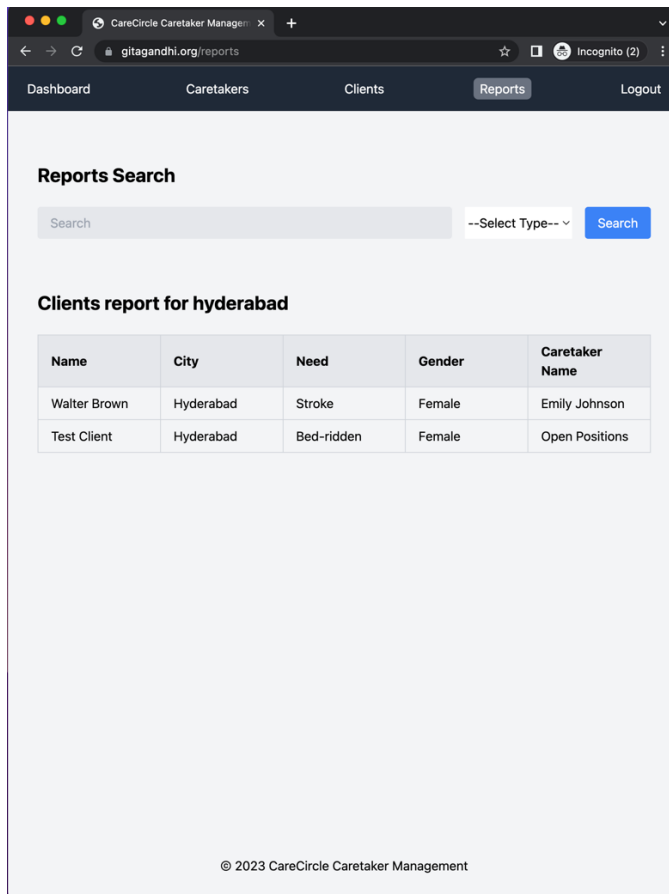
Clients report for all

Name	City	Need	Gender	Caretaker Name
Dorothy Harris	Kolkatta	Cancer	Male	Michael Davis
Alice Green	Delhi	Post-Surgery	Female	Jane Doe
Walter Brown	Hyderabad	Stroke	Female	Emily Johnson
Test Client	Hyderabad	Bed-ridden	Female	Open Positions
Henry Miller	Kolkatta	Diabetes	Male	Open Positions
Arthur Clark	Delhi	Bed-ridden	Male	Open Positions

© 2023 CareCircle Caretaker Management

Specific Columns for Clients

This report enables you to select specific columns related to clients, displaying only the chosen information.



Panopto Video Link

Name: D424CapstoneCMWA_WD-Task3

Link:

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=d04d8e78-14ce-4aac-948c-afd80159a9ac>