



# **CHAMP Human Performance Lab Data Management System**

James Petkin  
petkin.james@gmail.com

## Contents

1. Overview.....	3
1.1 Current Process .....	3
1.1.1 Issues.....	3
2. Background.....	3
2.1 Project Perspective.....	3
2.2 Intended Use .....	3
2.3 User Roles.....	3
2.3.1 Admin.....	3
2.4 Project Functions .....	4
3. Use Cases.....	4
3.1 Login/Logout .....	4
3.1.1 Diagram.....	4
3.1.2 User Logs In or Logs Out of the System .....	5
3.2 Uploading Data to The Database .....	6
3.2.1 Diagram.....	6
3.2.2 Uploading Data to The Database .....	6
3.3 Generating a Report .....	7
3.3.1 Diagram.....	7
3.3.2 Generating a Report for the Client.....	7
3.4 Viewing a Report (Ready Only) .....	8
3.4.1 Diagram.....	8
3.4.2 Viewing a Report .....	9
4. System Architecture .....	10
5. UI Design .....	11
5.1 Home Page .....	11
5.2 Login Page .....	12
5.3 Menu Navigation .....	13
5.4 Data entry/upload Page .....	14
5.5 Report Builder Page.....	15
5.6 Report Viewer Page .....	16
6. Database Design.....	17

6.1 MongoDB schemas .....	17
6.1.1 Users Collection.....	17
6.1.2 Report Collection .....	17
6.1.3 VO2max Test Collection.....	18
6.1.4 authUsers Collection.....	20
7. Non-Functional Specifications.....	21
7.1 Performance Requirements.....	21
7.2 Scalability .....	21
7.3 Security .....	21
7.4 Reliability & Availability .....	21
7.5 Maintainability .....	21
7.6 Usability .....	22
7.7 Portability.....	22
7.8 Data Integrity .....	22
8. Future Work .....	22
8.1 Expand Support for Additional Tests .....	22
8.2 Implement Test Filtering Capabilities.....	22
8.3 Migrate to a Full-Stack Web Framework.....	22
8.4 More Editability in the Report .....	22

# 1. Overview

## 1.1 Current Process

At the moment the CHAMP Human Performance Lab currently runs the tests, manually creates the report and manually runs their own analysis on the data. (eyeballing)

### 1.1.1 Issues

Through the discussion with the CHAMP head, there are a few things that jump out to us.

- The process is very manual. (Creation, Analysis, Presentation)
- The computers are old. (No internet access)
- Hard to keep data stored safely and securely. (Only stored in a USB stick, or on the computer the test was taken)
- No cloud storage so people or personnel can access the reports online. (All local)
- Very time-consuming process. (Takes a few hours)

## 2. Background

The CHAMP Human Performance Lab Data Management System aims to modernize the data management process for CHAMP by streamlining the storage, processing, and visualization of physiological test data. This system will enhance efficiency, improve data accessibility, and automate result interpretation.

### 2.1 Project Perspective

This system is a new software solution that will replace existing manual processes used for storing and analyzing test data. It will integrate with cloud-based platforms such as AWS and MongoDB Atlas to ensure scalability, security, and high availability.

### 2.2 Intended Use

This application is specifically designed for use by the CHAMP staff within the CHAMP Human Performance Lab here at SCSU. Thus, the application will be tailed according to the needs of the faculty within the department.

### 2.3 User Roles

#### 2.3.1 Admin

This role will be the only and highest-level user role within the new application. This is reserved for CHAMP personnel.

This role will:

- Allow administrators who collect and process data to access the uploader.
- Need access to raw and processed data for analysis in the report builder and viewer.

## 2.4 Project Functions

The system will facilitate:

- Uploading of data from the various tests done
- Secure data storage in the cloud
- Automated processing and visualization of test results
- Allow for minor adjustments if necessary\* (future)
- Allow the staff to filter specific tests on various dates and personnel\* (future)
- Allow staff to run statistics on the data stored\* (future)
- Integration with AWS and MongoDB Atlas for scalable and reliable data management
- Allow for download of completed report to either send or show to the client

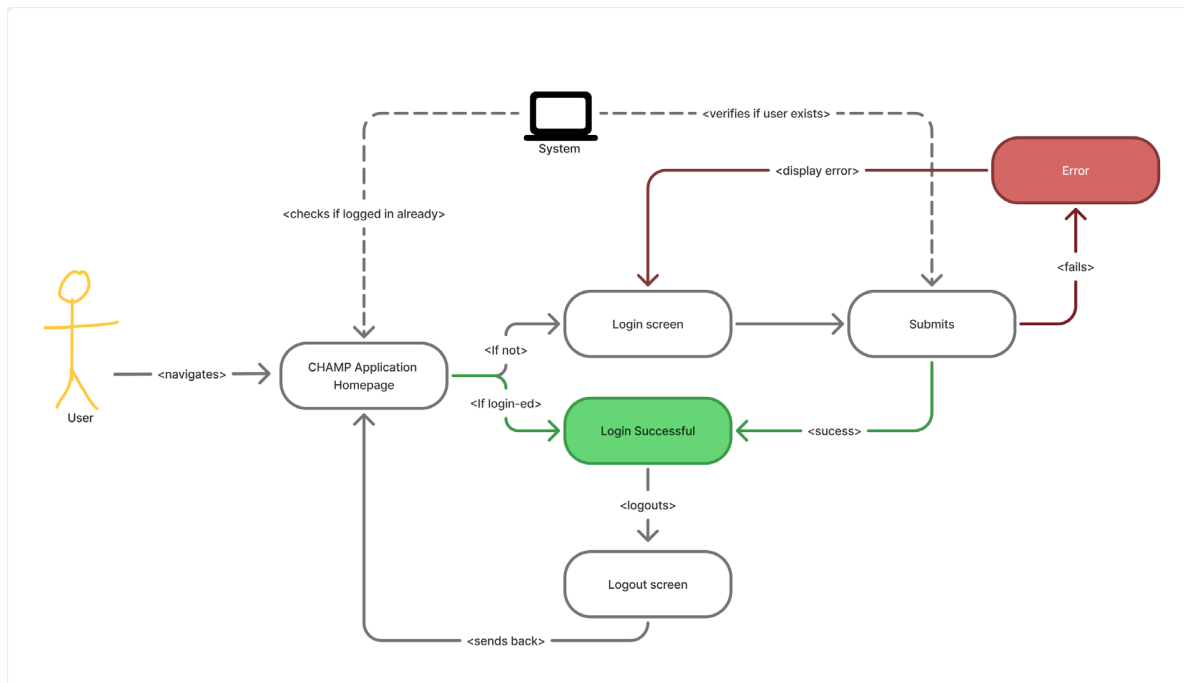
## 3. Use Cases

Some examples of the use cases of this application will be outlined below with accompanying Use Case diagrams.

### 3.1 Login/Logout

This use case describes how the admin will login to use the system or to logout if needed.

#### 3.1.1 Diagram



### 3.1.2 User Logs In or Logs Out of the System

#### 3.1.2.1 *Successful: Login*

- User navigates to the website.
- System checks if the user is already logged in.
- User is already logged in.
- System allows the user to navigate the application/menu.

OR

- User is not logged in.
- System redirects the user to the login screen.
- User enters their login credentials (e.g. email and password).
- System checks the login information against the database.
- Login credentials are valid.
- System grants access and logs the user in.
- User is allowed to navigate the application/menu.

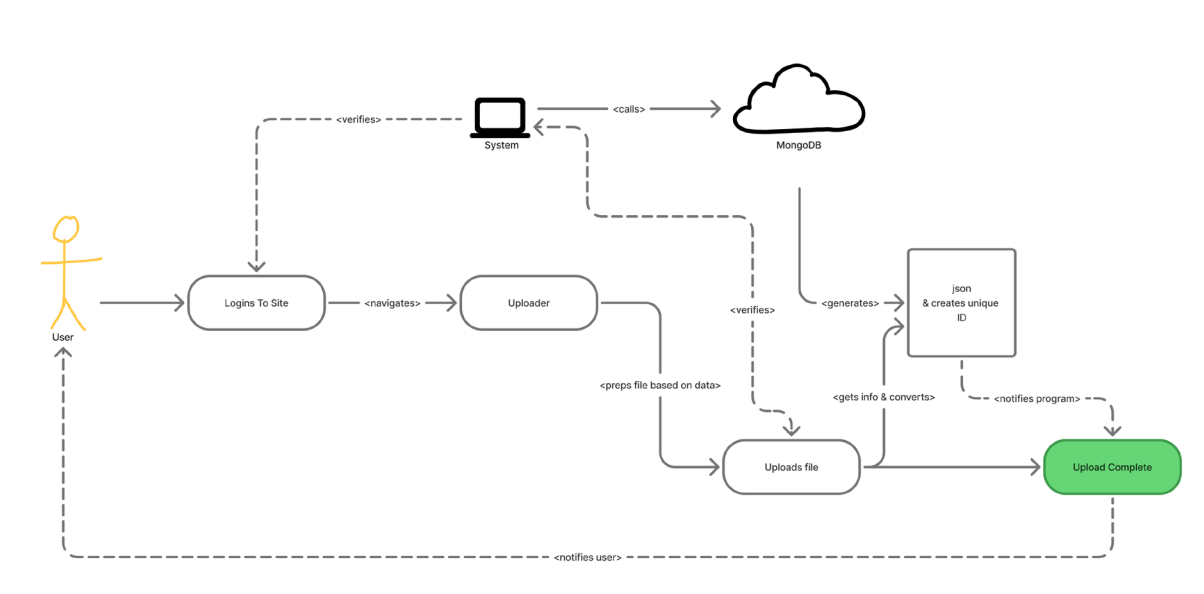
#### 3.1.2.2 *Unsuccessful: Login Failure*

- User navigates to the website.
- System checks if the user is already logged in.
- User is not logged in.
- System redirects the user to the login screen.
- User enters invalid login credentials (e.g. wrong password).
- System checks the login information against the database.
- Login information is invalid.
- System sends an error message to the user (e.g. “Invalid username or password”).
- User is prompted to try logging in again.

## 3.2 Uploading Data to The Database

This use case describes how the user would upload data to the database via the uploader on the web application.

### 3.2.1 Diagram



### 3.2.2 Uploading Data to The Database

#### 3.2.2.1 Successful: File is uploaded

- User navigates to the website.
- User logs in with valid credentials.
- System verifies login credentials.
- User redirect themselves to the uploader page.
- User selects a file to upload.
- System prepares the file based on the data.
- System verifies credentials with MongoDB before upload.
- System converts the file data to JSON.
- File is uploaded to MongoDB.
- MongoDB generates a unique ID for the uploaded data.
- MongoDB stores the JSON in the database.
- Upload completes successfully.
- System notifies the user that the upload is complete.

#### 3.2.2.2 Unsuccessful: File does not upload

- User logs in successfully.
- User is redirected to the uploader page.
- User selects a file to upload.

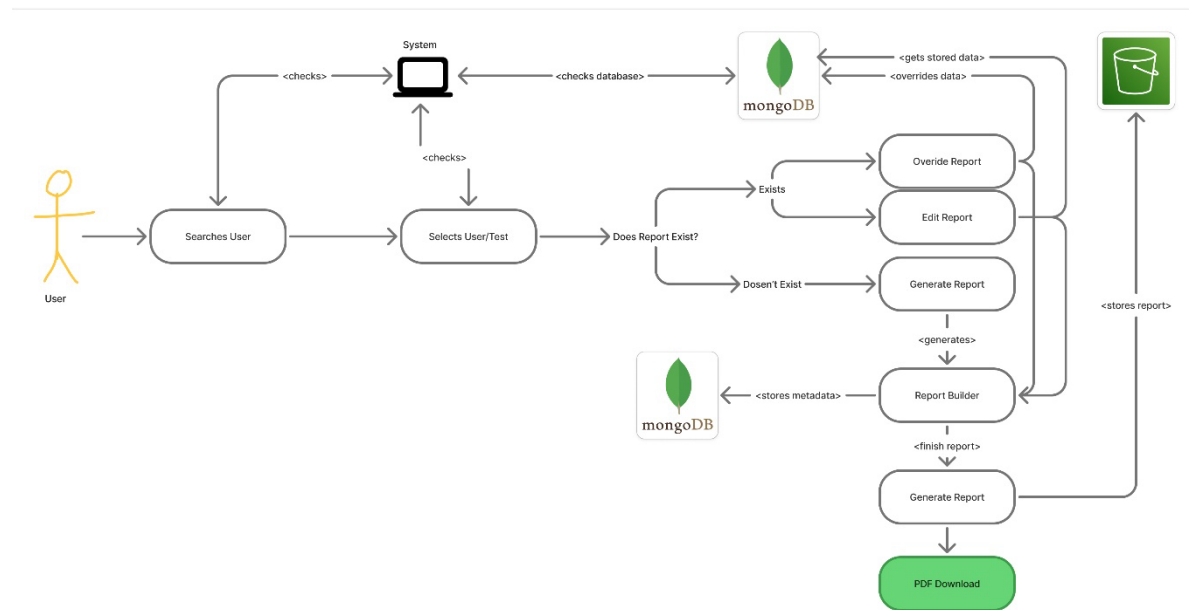
- System attempts to prepare or verify the file.
- File format is invalid, or data fails verification.
- Upload process is stopped.
- File is not stored in the database.
- System notifies the user that the upload failed with an appropriate error message.

### 3.2.2.3 Unsuccessful: Login Fails

- User navigates to the website.
- User enters incorrect login credentials.
- System rejects the login attempt.
- User is shown an error message.

## 3.3 Generating a Report

### 3.3.1 Diagram



### 3.3.2 Generating a Report for the Client

#### 3.3.2.1 Successful: User is found

- User searches for a client in the system.
- System successfully finds and displays the matching user.

#### 3.3.2.2 Successful: User has test

- After selecting the user, the system confirms the user has test data available in the database.



### 3.3.2.3 Successful: Generating a report for a client

- System checks if a report exists:
  - If yes, user may edit or override it.
  - If no, system generates a new report.
- MongoDB provides test data and metadata.
- Report Builder compiles the information.
- Final report is created and stored in cloud storage (e.g., AWS S3).
- User downloads the completed PDF.

### 3.3.2.4 Unsuccessful: User not found

- User searches for a client, but no matching record exists in the database.
- System displays an error: “User not found.”

### 3.3.2.5 Unsuccessful: User has no test

- User is found, but no associated test data exists in the database.
- System displays an error: “No test data found for this user.”

### 3.3.2.6 Unsuccessful: Report is not generated (due to backend failure)

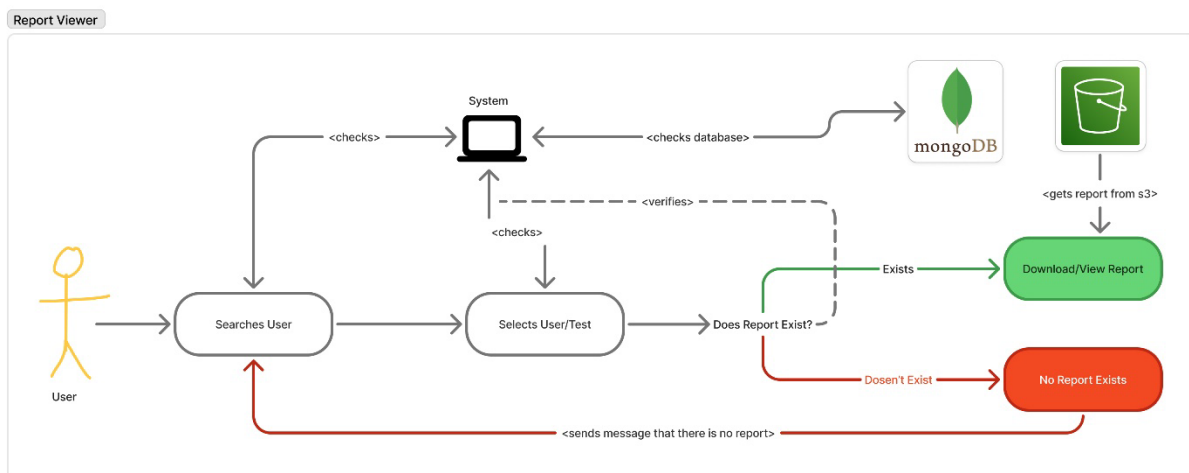
- System fails to retrieve required data or metadata from MongoDB.
- Generation fails and the user is notified: “Unable to generate report. Please check data connection or content.”

### 3.3.2.7 Unsuccessful: Report Builder fails

- Report Builder encounters missing values or formatting errors during creation.
- Report generation is aborted, and the system shows: “Report generation failed due to formatting or data issues.”

## 3.4 Viewing a Report (Ready Only)

### 3.4.1 Diagram



## 3.4.2 Viewing a Report

### 3.4.2.1 *Successful: User is found*

- User searches for a client in the system.
- System successfully finds and displays the matching user.

### 3.4.2.2 *Successful: User has report documented*

- After selecting the user, the system verifies that report data exists.

### 3.4.2.3 *Successful: Report exists and is viewable*

- System checks if a report is available in MongoDB and/or AWS S3.
- Report exists.
- System retrieves the report from S3 storage.
- User is able to view or download the report.

### 3.4.2.4 *Unsuccessful: User not found*

- User enters search query.
- System cannot find any user matching the query.
- System displays an error: “User not found.”

### 3.4.2.5 *Unsuccessful: User has no report documented*

- User is found but there is no report data associated.
- System informs the user: “No report data found for this user.”

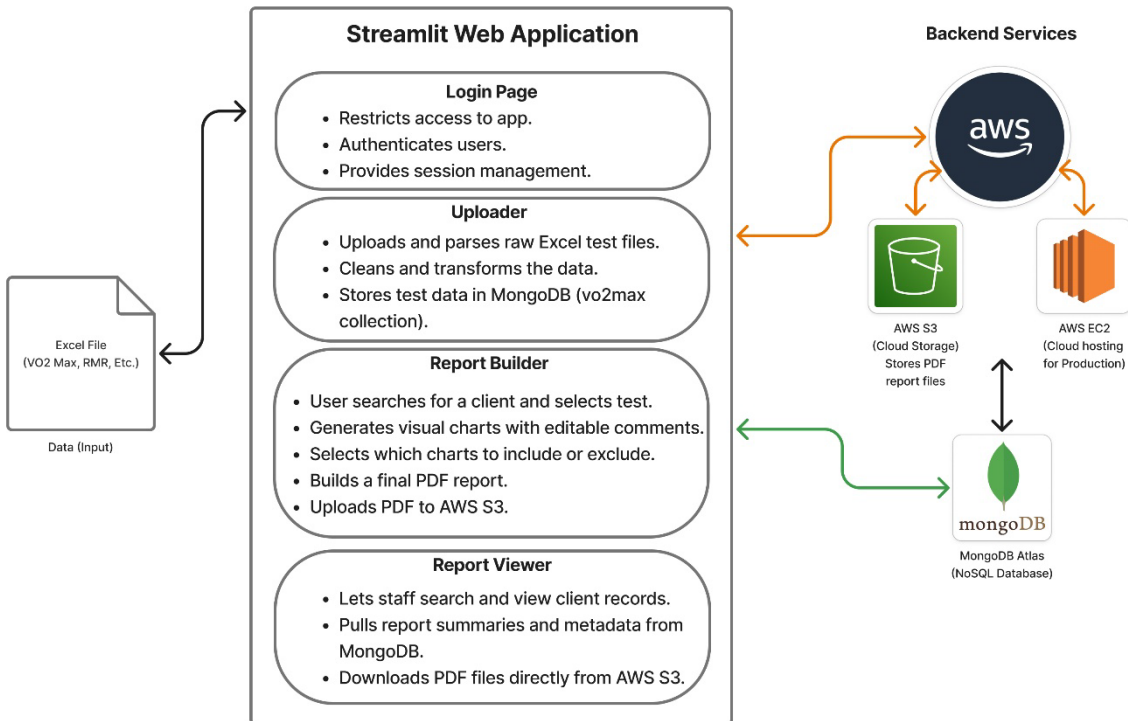
### 3.4.2.6 *Unsuccessful: Report does not exist*

- System checks for a report but does not find one in the database or S3.
- User is informed: “No report exists.”

### 3.4.2.7 *Unsuccessful: Report retrieval fails*

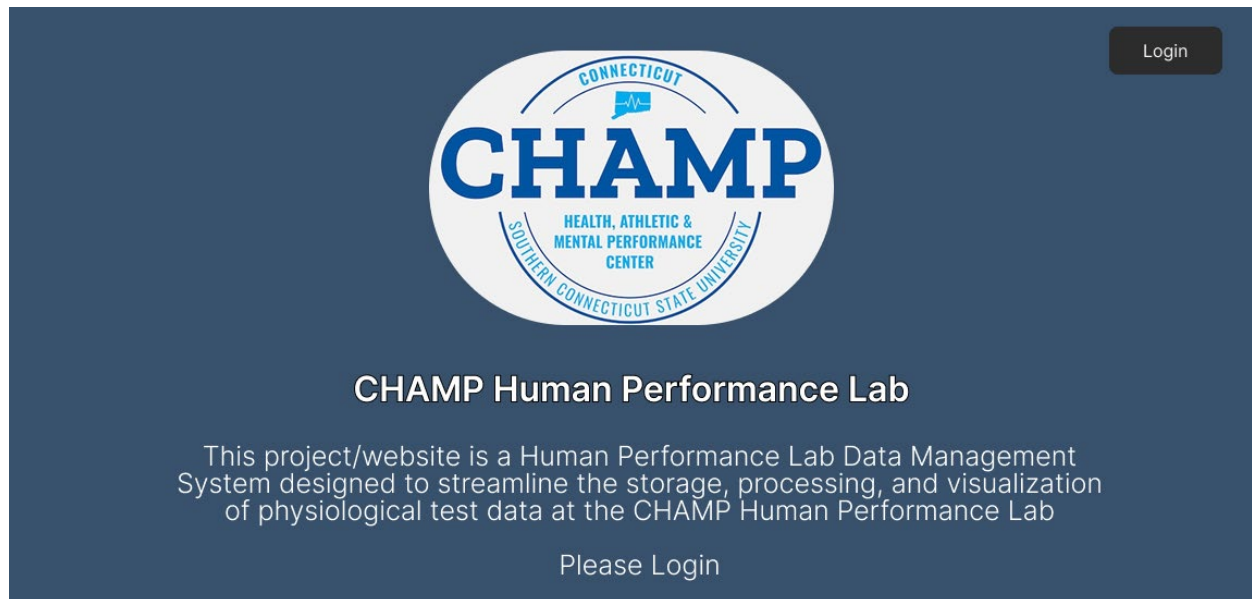
- A report exists, but system fails to fetch it from AWS S3 (e.g., due to a connection error).
- System displays an error message: “Unable to load report. Please try again later.”

## 4. System Architecture



## 5. UI Design

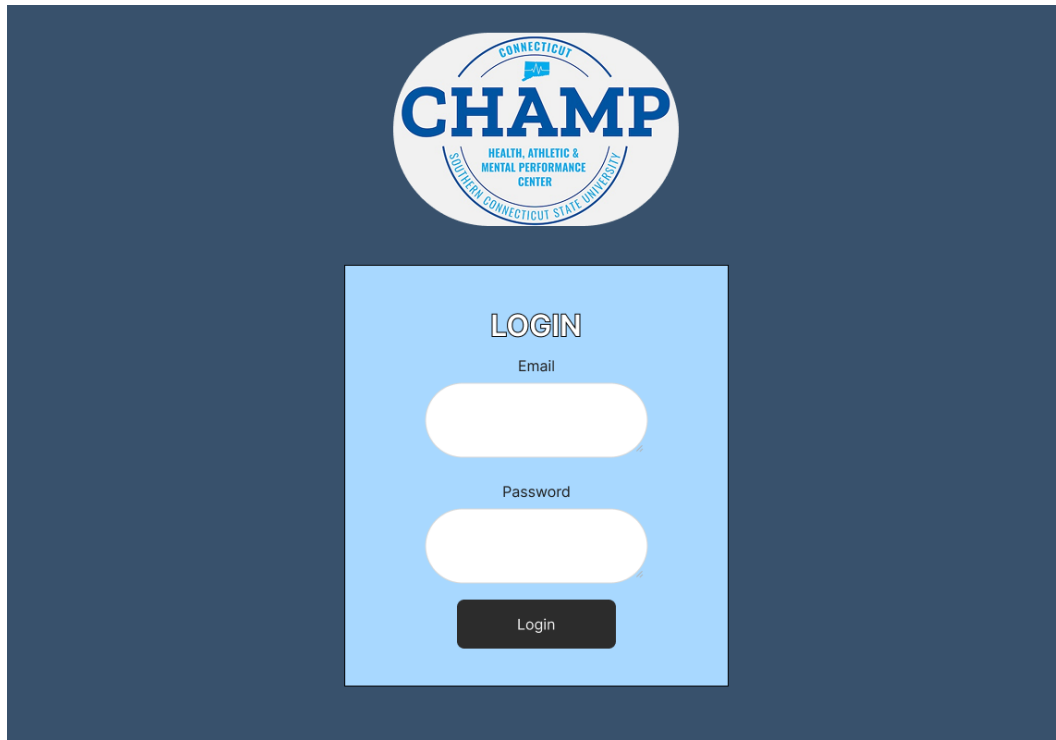
### 5.1 Home Page



Includes:

- Login/Logout button that leads the user to the login page.
- Menu is hidden until the user is authenticated and login.

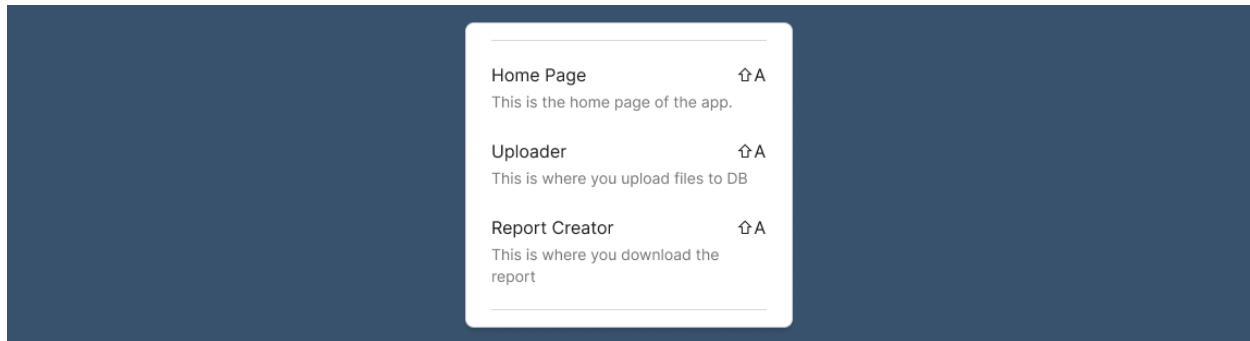
## 5.2 Login Page



Includes:

- Login screen where the user can login with email and password.
- Once user logs in, it will take them back to the home screen with the menu icon unhidden.

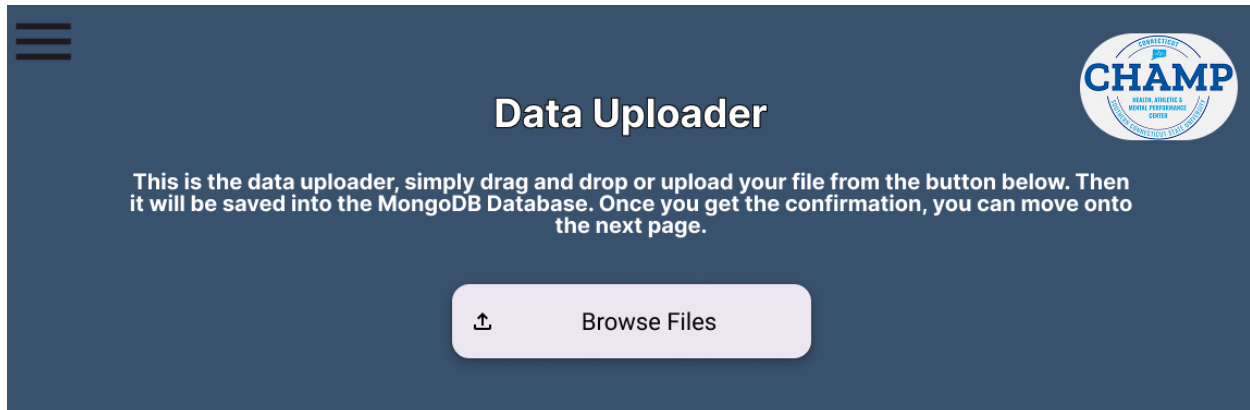
## 5.3 Menu Navigation



Includes:

- All the current pages the user can navigate to. (Based on authorization of user)
- Allows user to navigate to the uploader or the report creator.

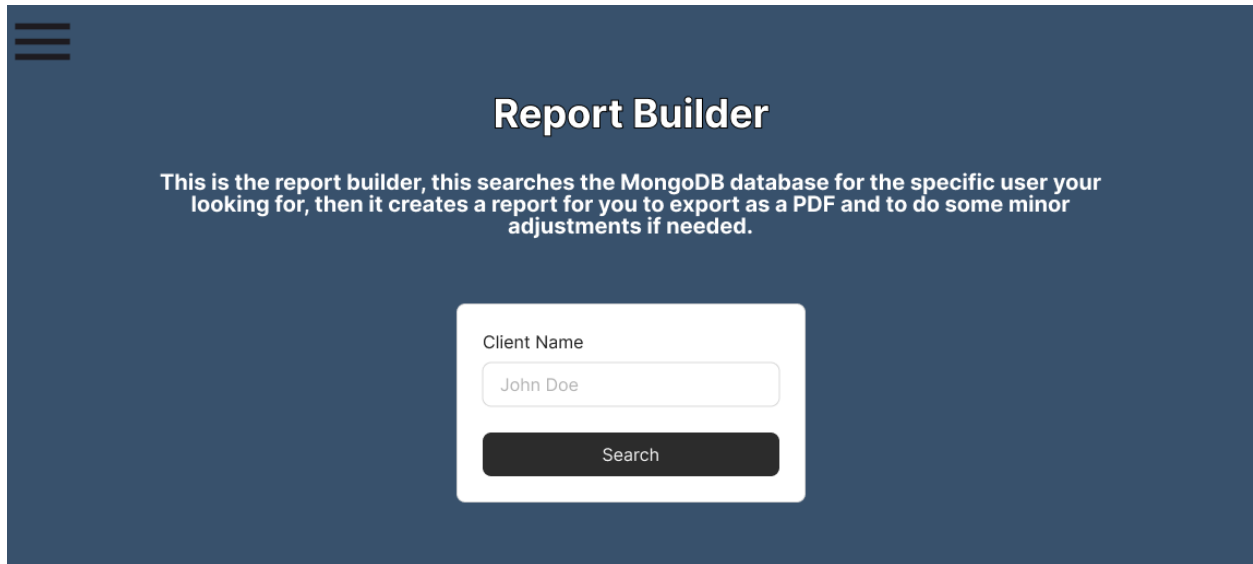
## 5.4 Data entry/upload Page



Includes:

- Data uploader allows the user to upload data for the MongoDB database. (In the cloud)
- Stores the data in the cloud allowing the user to go to the Report Creation page to generate a report.

## 5.5 Report Builder Page

The screenshot shows a web interface for a 'Report Builder'. It has a dark blue background. In the top left corner, there is a hamburger menu icon consisting of three horizontal lines. The title 'Report Builder' is centered in a large, white, sans-serif font. Below the title, a paragraph of white text explains the function: 'This is the report builder, this searches the MongoDB database for the specific user your looking for, then it creates a report for you to export as a PDF and to do some minor adjustments if needed.' In the center of the page, there is a white rectangular form. Inside this form, the text 'Client Name' is positioned above a text input field. The input field contains the text 'John Doe'. Below the input field is a dark grey button with the word 'Search' in white text.

Report Builder

This is the report builder, this searches the MongoDB database for the specific user your looking for, then it creates a report for you to export as a PDF and to do some minor adjustments if needed.

Client Name

John Doe

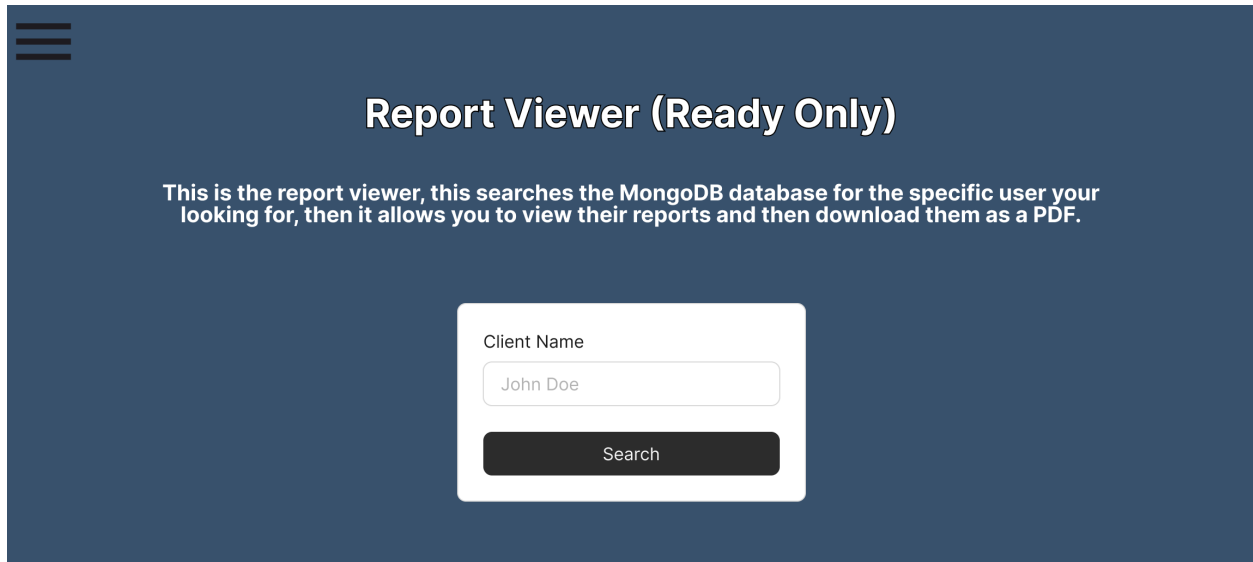
Search

Includes:

- Searches for MongoDB for a specific client test data based on the provided name.
- If found, the system retrieves the data and allows option to generate a structured report.
- The user can preview, make small adjustments, and export the report as a PDF.
- Allows seamless transition from data management to report delivery in one interface.



## 5.6 Report Viewer Page



The image shows a web interface for a report viewer. It has a dark blue background. In the top left corner, there is a hamburger menu icon consisting of three horizontal lines. The main heading is "Report Viewer (Ready Only)" in a large, bold, white font. Below the heading, there is a paragraph of text in white: "This is the report viewer, this searches the MongoDB database for the specific user your looking for, then it allows you to view their reports and then download them as a PDF." In the center of the page, there is a white rectangular box containing a search form. The form has a label "Client Name" above a text input field. The input field contains the text "John Doe". Below the input field is a dark grey button with the word "Search" in white text.

Includes:

- Searches MongoDB for a client by name to locate previously generated reports.
- Provides a read-only interface where users can view report contents without making changes.
- Enables downloading the report as a PDF for sharing or documentation.

## 6. Database Design

### 6.1 MongoDB schemas

#### 6.1.1 Users Collection

Field Name	Type	Description
id	ObjectId	Unique identifier automatically generated by MongoDB.
name	String	Patient's full name (e.g., "JOHN, DOE").
age	Integer	Patient's age in years (e.g., 42).
sex	String	Patient's biological sex (e.g., "F" for female, "M" for male).
fileNumber	String	Unique patient file number as found in original reports (e.g., "1234").
height	Integer	Patient's height (in centimeters).
weight	Integer	Patient's weight (in pounds).
doctor	String	Initials or name of the supervising doctor associated with the test (e.g., "DGM").
testIds	Array<ObjectId>	Array of references to documents in the tests collection representing associated test records.

#### 6.1.2 Report Collection

##### General

Field Name	Type	Description
_id	ObjectId	Unique identifier for the report document.
test_id	ObjectId	Reference to the associated test document.
user_id	ObjectId	Reference to the user who took the test.
last_updated	DateTime	Timestamp of the last report update.

##### Plots (Array of Objects)

Field Name	Type	Description
index	Integer	Position of the plot in the array.
title	String	Title of the plot (e.g., 'V-Slope').
comment	String	Optional comment for the plot.

include	Boolean	Flag indicating whether to include the plot in the final report.
---------	---------	--

#### Summary

Field Name	Type	Description
summary	String	Optional summary or interpretation text associated with the report.

#### Test Date

Field Name	Type	Description
Year	Integer	Year the report/test was conducted.
Month	Integer	Month the report/test was conducted.
Day	Integer	Day the report/test was conducted.

### 6.1.3 VO2max Test Collection

#### Report Info

Field Name	Type	Description
School	String	Name of the school or organization conducting the test
Date.Year	Integer	Year of the test
Date.Month	Integer	Month of the test
Date.Day	Integer	Day of the test
Time.Hour	Integer	Hour of the test
Time.Minute	Integer	Minute of the test
Time.Second	Integer	Second of the test

#### Patient Info

Field Name	Type	Description
File Number	String	Client file number
Name	String	Client name
Age	Integer	Client age
Height	Integer	Client height
Sex	String	Client sex
Weight	Integer	Client weight

Doctor	String	Supervising doctor
--------	--------	--------------------

## Test Protocol

Field Name	Type	Description
Test Degree	String	Test difficulty level
Exercise Device	String	Device used for exercise

## Test Environment

Field Name	Type	Description
Insp. Temp	Float	Inspired temperature
Baro. Pressure	Float	Barometric pressure
Insp. humid	Float	Inspired humidity
Exp. flow temp.	String	Expired flow temperature
Insp. O2	Float	Inspired O2 concentration
Insp. CO2	Float	Inspired CO2 concentration
Selc. Flowmeter	String	Selected flowmeter
STPD to BTPS	Float	STPD to BTPS conversion factor
O2 Gain	Float	Oxygen gain setting
CO2-NL gain	Float	CO2-NL gain setting

## Best Sampling Values

Field Name	Type	Description
Base O2	Float	Baseline O2
Base CO2	Float	Baseline CO2
Measured O2	Float	Measured O2
Measured CO2	Float	Measured CO2

## Results

Field Name	Type	Description
Max VO2	Float	Maximum VO2
VO2max Percentile	Float (optional)	Percentile ranking of VO2 max

## Tabular Data

Field Name	Type	Description
Time	Float	Measurement time point
VO2 STPD	Float	VO2 in STPD
VO2/kg STPD	Float	VO2/kg in STPD
Mets	Float	Metabolic equivalents
VCO2 STPD	Float	VCO2 in STPD
VE BTPS	Float	Minute ventilation in BTPS
RER	Float	Respiratory exchange ratio
RR	Float	Respiratory rate
Vt BTPS	Float	Tidal volume in BTPS
FEO2	Float	Fraction of expired O2
FECO2	Float	Fraction of expired CO2
HR	Float	Heart rate
TM SPD	Float (optional)	Treadmill speed
TM GRD	Float (optional)	Treadmill grade
AcKcal	Float	Accumulated kilocalories
PetCO2	Float	End-tidal CO2
PetO2	Float	End-tidal O2
VE/VCO2	Float	Ventilation to CO2 ratio
VE/VO2	Float	Ventilation to O2 ratio
FATmin	Float	Fat oxidation minimum
CHOmin	Float	Carbohydrate oxidation minimum

### 6.1.4 authUsers Collection

Field Name	Type	Description
_id	ObjectId	Unique identifier for the user (automatically generated by MongoDB).
username	String	The user's login name.
password	Binary	Base64-encoded hashed password for authentication.

## 7. Non-Functional Specifications

### 7.1 Performance Requirements

- The system should respond to user actions (e.g., login, file upload, report view) within 10 seconds under normal load.
- Report generation should complete in less than 15 seconds for standard-sized files.
- MongoDB queries and S3 fetches must be optimized for fast I/O to avoid lag in report viewing.

### 7.2 Scalability

- The system must be capable of supporting an increasing number of users and upload report metadata.
- MongoDB Atlas and AWS S3 should scale automatically to accommodate data growth without degrading performance.

### 7.3 Security

- Passwords must be stored securely using hashing and binary encoding (as shown in the `authUsers` schema).
- User access must be restricted by authentication; only logged-in users can upload, view, or generate reports.
- Communication with the database and cloud storage must use encrypted protocols (e.g., HTTPS, SSL/TLS).

### 7.4 Reliability & Availability

- The system must be available 99% of the time during lab hours.
- In case of backend failure (e.g., MongoDB or S3 unavailability), meaningful error messages must be displayed to the user.
- Data uploads and reports should be atomic — either they fully complete or fail gracefully without corrupting data.

### 7.5 Maintainability

- The backend logic is modularized (e.g., uploader, viewer, report builder) to simplify maintenance and future updates.
- Code should follow consistent formatting and include inline comments for clarity.

## 7.6 Usability

- The interface is user-friendly with clear prompts and visual feedback.
- Error messages should help the user understand what went wrong (e.g., "No report found", "Invalid file format").

## 7.7 Portability

- The web application must be accessible on modern desktop browsers (Chrome, Firefox, Edge).

## 7.8 Data Integrity

- Test results and reports must be stored without modification once generated.
- Unique identifiers (IDs) ensure traceability across users, reports, and tests.

# 8. Future Work

## 8.1 Expand Support for Additional Tests

- Extend the system to handle other physiological assessments such as Resting Metabolic Rate (RMR), Lactate Threshold, Gait Analysis, and many others using the same data pipeline and report structure established for VO2 Max.

## 8.2 Implement Test Filtering Capabilities

- Implement a filtering system to allow staff to view client progress over custom timeframes. This would enhance usability, especially as the dataset grows.

## 8.3 Migrate to a Full-Stack Web Framework

- Transition from Streamlit to a full-stack web development approach using HTML, CSS, JavaScript, and a backend framework (e.g., Flask or Node.js). This would improve UI customization, frontend flexibility, and scalability for production deployment.

## 8.4 More Editability in the Report

- Enable editable data visualizations, allowing threshold adjustments and value customization.

