**Data Getta Development Docs**

**AI for Auburn University Baseball - Database**

**Developer Documentation**

Authors:

Teddy Cooke

Laura Cromwell

Frankie Donze

Caden Garrett

Micah Key

Braden Mosley

Jacob Munroe

Annie Murphy

COMP 4710: Senior Design Team #9 Orange and Blue

Department of Computer Science and Software Engineering

Samuel Ginn College of Engineering, Auburn University

**Table of Contents**

[**Developer Docs for the server Infrastructure 3**](#_qnlvs7hiban)

[Server General Information 3](#_da114lum2smt)

[Documentation 3](#_uyc72bt7rxlq)

[Server and Project Access 3](#_1srxaw89xd6)

[Cron Script 4](#_it0fumcy951l)

[Future Work and Goals 4](#_p4ph04h6b6s4)

[**Developer Docs for the Postgres Database 5**](#_f3eo8ujjeulx)

[Database General Information 5](#_oyrhmpac5mcy)

[Version 5](#_e3pz9jl94osu)

[Documentation 5](#_iequ3am5yf46)

[Future Work and Goals 6](#_pzww2zch5leu)

[**Developer Docs for The Eye - A Data Getta Product for Auburn Baseball 7**](#_tf8zrz9xgivf)

[Domain 7](#_k60slyfdsig2)

[Software Utilized 7](#_q9hzl58caez7)

[Next.js 7](#_9tgiaaoy0spb)

[Material UI 8](#_1kwi6u5k4rvm)

[Prisma 9](#_jewpejn0l2my)

# 

# Note: Most recent versions of all development docs are on the [Data Getta GitHub](https://github.com/bradenmosley/datagetta).

# Developer Docs for the server Infrastructure

Last Updated: April 22, 2024 by Caden Garrett

## Server General Information

The server for the Data Getta project is setup and ran by the OIT department at Auburn University, and is ran on campus and is accessible through the Auburn University VPN. It is a Linux machine and is running Ubuntu 22.04 LTS. The specs of the server are as follows:

24 GB of RAM

2 CPUs

5 TB of storage

Current URL is datagetta.cse.eng.auburn.edu

This server is used to host the Postgresql database, the Next.js frontend, and the Python code for the modeling teams. All other code associated with the Data Getta project is stored on the server as well.

## Documentation

Here are a few links that are useful for managing the server:

Ubuntu Documentation - <https://ubuntu.com/server/docs>

Docker Documentation - <https://docs.docker.com/>

Bash Documentation - <https://www.gnu.org/software/bash/manual/bash.html>

## Server and Project Access

Since OIT manages the machine, the way to access the server if you are a new developer on this project is to email a member of OIT and request an account. The account will be created and you will be given the username and password to access the server under your own account.

Currently, we have most of the code available via a git pull from our GitHub repository. However, there are some env files and other configurations that was intentionally left out of the public repo. These files will need to be copied over to your new account on the server. The current location of the Data Getta project is in the /home/csg0026/project/datagetta folder.

**NOTE: DO NOT CHANGE SSH SETTINGS, ACCOUNT NAMES, OR PASSWORDS WITHOUT PERMISSION FROM OIT.**

Project Configuration

Since we are using different services for our UI, backend, and python code, we decided to use docker-compose to manage all of the services. This allows us to easily start, stop, and manage all of the services at once.

The current docker-compose.yml file is located in the root of the project. This file is used to start all of the services that are needed for the Data Getta project. The services that are currently being used are:

* Postgresql
* Next.js
* Python
* The docker-compose.yml file is used to start all of the services at once. To start the services, you will need to run the following command:

docker compose up -d --build

This will start all of the services and you will be able to access the UI at datagetta.cse.eng.auburn.edu.

## Cron Script

The server has a cron script that is used to add new data and update the whole project afterwards. This script is set to run every Monday morning at 2:00 AM CST. This is done to ensure that the new data is updated and the whole project is updated before the start of the work week. Currently, the script does nothing of use but the setup is there to easily add the functionality.

## Future Work and Goals

Add the ability to automatically create backups of the data and store them in a secure location.

Finish the cron script to add new data and update the project.

# Developer Docs for the Postgres Database

Last Updated: April 22, 2024 by Caden Garrett

## Database General Information

The database for the Data Getta project is a Postgresql Docker container hosted on the Data Getta server. The current username, password, and port are all located in the db.env file in the root of the project. The current port that is used for connecting to the database is the default port of 5432.

## Version

The current version of Postgresql that is being used is Version 16.

## Documentation

Here is the link for the official Postgresql documentation: https://www.postgresql.org/docs/

Here is also a link for learning SQL: https://www.w3schools.com/sql/

**Database SQL Files**

The way that SQL scripts are stored and ran is through a series of SQL files in the database/sql folder in the root of the project. The reason for this is to allow for easy version control and to allow for easy access to the SQL scripts and a easy way to run them if something were to happen to the database. Here is a current list of the SQL files in the database/sql folder and what they do:

add-conferences.sql - Adds the NCAA Division 1 baseball conferences to the database in the conferences table.

functions.sql - Contains all of the functions that are used both by the database itself and the modeling teams connecting to the database. Documentation of the functions and what they do individually can be found within the actual file.

keepd1.sql - When ran, will go through and delete all teams that have their conference set to NotSet and every bit of data that is associated with those teams. This is ran immediately after the team-assignment.sql script is ran.

pre-schema.sql - Provides the database with all the necessary extensions, types, or anything else that is needed before the schema is created.

schema.sql - Houses the schema used for the database. This includes all of the tables, constraints, and indexes that are used in the database.

seasons.sql - Adds the NCAA Division 1 baseball seasons to the database in the seasons table.

team-assignment.sql - Assigns each team to a conference based on the conference that is in the conferences table. This script is intended to be run after any new data is added to ensure all the correct and relevant data is present in the database.

views.sql - Contains all of the views that are used by the database itself and the modeling teams connecting to the database. Documentation of the views and what they do individually can be found within the actual file.

## Future Work and Goals

None.

# Developer Docs for The Eye - A Data Getta Product for Auburn Baseball

Last Updated: April 21, 2024 by Braden Mosley

## Domain

<https://datagetta.app>

The domain is purchased using Squarespace. Sign in using the Google account with the email auburndatagetta@gmail.com . The password is located at \_\_\_ on the Data Getta server.

The domain had to be setup to "point to" datagetta.cse.eng.auburn.edu to allow the utilization of Auburn's authentication service.

The domain will auto-renew on February 5, 2025.

## Software Utilized

Next.js (App Router)

React

Material UI

Prisma

Docs

Next.js - <https://nextjs.org/docs>

React - <https://react.dev/reference/react>

Material UI - <https://mui.com/material-ui/all-components/>

Prisma - <https://www.prisma.io/docs>

## Next.js

It is highly recommended to become familiar with Next.js before beginning development on this project.

Key topics to begin with:

File structure

https://nextjs.org/docs/app/building-your-application/routing/defining-routes

Layout.tsx vs Page.tsx

https://nextjs.org/docs/app/building-your-application/routing/pages-and-layouts

https://nextjs.org/docs/app/api-reference/file-conventions/layout

https://nextjs.org/docs/app/api-reference/file-conventions/page

Server vs Client Components

https://nextjs.org/docs/app/building-your-application/rendering/server-components

https://nextjs.org/docs/app/building-your-application/rendering/client-components

Dynamic Routes

https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes

Folder Structure

All of the user interface code is located inside the /app folder.

Inside the /app folder is the layout and page for the landing page and the /the-eye folder.

The /the-eye folder houses the majority of the UI code. Putting everything inside of this folder allows for authentication to redirect you to datagetta.app/the-eye once logged in.

Inside the /the-eye folder, there is a folder for player related pages, team related pages, and the UI layout components.

Inside the /player and /team folder are a series of dynamic route folders that are used to query the database and fill out each of the corresponding pages.

Dynamic Routes

Dynamic routes are utilized in this project to query the database. When passing info like player names and team names in dynamic routes, you will need to utilize the built-in function decodeURIComponent(). This function is used to handle special characters (like spaces) passed in a url.

For example:

If the url passed is domain.com/param1 param2

The url will render as domain.com/param1%20param2

The function will replace the %20 with a space

Authentication

Authentication is handled using Auburn's authentication service.

## Material UI

Material UI was chosen as a way to ensure and provide standard styling across the user interface. This is essential as this project will be passed from group to group. Material UI also provides a lot of functional components that are taken advantage of in this project (date pickers, bar graphs, tables, etc...).

**WHEN DEVELOPING ON THIS PROJECT, PLEASE MAKE USE OF ALL OF MATERIAL UI'S COMPONENTS TO ENSURE CONSISTENT STYLING ACROSS THE UI.**

For example, use Material UI's Typography component instead of using 'h1' - 'h6' and 'p' HTML elements.

Links

When needing to add a link to a page, a custom Link component is located in '/app/utils/Link.tsx'. This component combines Next's Link component and Material UI's Link component. With Next's Link component being the primary way to navigate between routes, combining it with the Material UI's Link component adds styling to the link.

## Prisma

Prisma is used to interact with the database. Prisma provides its own query syntax, allowing you to not have to be familiar with SQL and it will provide effecient queries out-of-the-box. Unfortunately, Prisma queries do not support database features like views and functions, so you will have to make use of 'prisma.$queryRaw' in these instances. (There are examples of both Prisma queries and raw SQL queries in this project.)

Prisma Schema

To be able to develop locally using Prisma, you will need a 'schema.prisma' file located in the /prisma folder. The 'schema.prisma' file is created at build time on the Next app from the schema written in SQL. If the schema is ever updated, the 'schema.prisma' file in the repository will become outdated and the newly created 'schema.prisma' file will have to be pushed from the server after the Next app is built. After pulling the newly updated prisma schema, you will have to locally run the command npx prisma generate to update your local Prisma client.

**DO NOT MAKE CHANGES TO THE SCHEMA FROM THE SCHEMA.PRISMA FILE. YOU WILL HAVE TO MAKE CHANGES TO THE SCHEMA IN THE SCHEMA.SQL FILE IN THE DATABASE DIRECTORY.**

Querying the Database

One issue that was encountered when querying the database was the Postgres BigInt type. Certain Material UI components did not like the BigInt type being passed into it so the BigInt type had to be converted to the Number type. This is done by using JSON.parse(JSON.stringify(data, replacer)) where 'data' is the object queried from the database and 'replacer' is the function that converts the BigInt type to the Number type. The replacer functions are located in /app/utils/replacer.ts .