VisionAid STATS DEVELOPER MANUAL

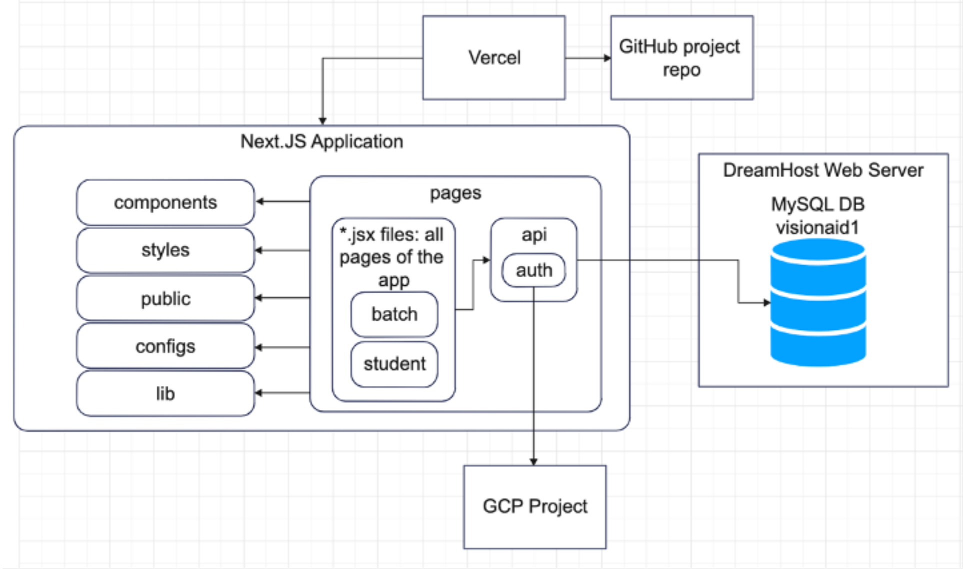
Georgia Tech OMSCS C4G, Spring 2023

PROJECT DESCRIPTION

The VisionAid STATS application enables Vision Aid staff to manage the courses offered by their program, batches for the courses, student enrollments in batches, and the grades and attendance for students in each batch.

ARCHITECTURE DIAGRAM

The VisionAid STATS application has a Next.JS frontend and a MySQL DB backend. It is deployed to AWS public cloud via Vercel and leverages GCP OAuth for authentication. As shown in the architecture diagram, the Vercel project and the GitHub repo are integrated such that GitHub automatically creates a preview deployment to test that there are no deployment errors upon every PR that is made to the GitHub repo. Vercel is also configured with environment variables specifying the credentials for the DB, so that the `api` module of the application can access it, as well as the GCP project `GOOGLE_CLIENT_ID` and `GOOGLE_CLIENT_SECRET` so that the application's `auth` module can verify authenticated users stored in the project.



All application logic resides in the `pages` directory, which includes all *.jsx files that render the pages of the app. Each page file includes api calls through methods defined in files in the `api` directory, which connect to the MySQL DB, make queries, and return results to the page. The page files also leverage various helper scripts, reusable components (ie: Buttons, NavBar), icons, and CSS styles from the `components`, `styles`, `public`, `configs`, and `lib` directories as shown in the architecture diagram.

DATA FLOWS

Here are the data flows for some of the key existing user stories:

User Stories	Data Flow
Add new staff member to system	1. Management user enters info for new staff member in the “Add New Staff Member” form 2. Form submission is handled in pages/users.jsx with a POST API call made to /api/usercreate 3. pages/api/usercreate.js takes form data, connects to visionaid1 database, and creates a new record in the vausers table with the data 4. users.jsx re-renders the page to fetch all the data with from vausers by making an API call to pages/api/getusersdata.js to show the newly entered data
Create new course	1. Management user enters info for new course in the “Create Course” form on the Courses page 2. Form submission is handled in pages/courses.jsx with a POST API call made to /api/coursecreate 3. pages/api/coursecreate.js takes form data, connects to visionaid1 database, and creates a new record in the vacourses table with the data 4. courses.jsx re-renders the page to fetch all the data from vacourses by making an API call to pages/api/getcoursedata.js to show the newly entered data
Create new batch for existing course	1. Management user enters info for new batch in the “Create Batch” form on the Batches page 2. Form submission is handled in pages/batches.jsx with a POST API call made to /api/batchcreate 3. pages/api/batchcreate.js takes form data, connects to visionaid1 database, and creates a new record in the vabatches table with the data 4. batches.jsx re-renders the page to fetch all the data from vabatches by making an API call to pages/api/getbatchdata.js to show the newly entered data
Register student in system	1. Management user enters info for new student in the form on the Student Registration page (under Students) 2. Form submission is handled in pages/studentregistration.jsx with a POST API call made to /api/studentapplication 3. pages/api/studentapplication.js takes form data, connects to visionaid1 database, and creates a new record in the vastudents table with the data 4. User navigates to Students page 5. students.jsx renders the page to fetch all the data from vastudents by making an API call to pages/api/getstudentsdata.js to show the newly entered data
Enroll a student in a batch	1. PM clicks on the “Roster” button from the table in the Batches page which redirects them to api/batch/[batch id] 2. On this page, they click the “Add Student” button which prompts them to select from a list of “Unassigned Students” to enroll them in the batch. The Unassigned Students list is fetched with the fetchUnassignedStudents method in pages/batch/[id].jsx and makes a POST API call to api/getunassignedstudents with a request body including the batch_id. The response is a list of student id’s and names from the vastudents table where

	the student id is not in the list of students in the vastudent_to_batch table that are already enrolled in the batch of that batch id. 3. The addStudent method in pages/batch/[id].jsx then is called taking the studentId of the unassigned student to be added as input and makes a POST API call to api/addstudenttobatch where a record gets created in the vastudent_to_batch table with the student id and batch it. New records are also created in the va_grades table assigning all the existing assignment names for the batch to the student. New records are also inserted in va_attendance with the student id, batch id, and all the dates for the batch with the default being set to false. 4. User should see new row for student in the Grades and Attendance tables on the Batch Roster page.
Enter attendance for a batch	1. User toggles from “Absent” to “Present” for attendance for a student on a particular date in the attendance table 2. updateAttendance method in pages/batch/[id].jsx is called with the batchId, studentId, date, and the new isPresent value as input. The method makes a POST API call to api/updateattendance with the request body including all the method inputs. The corresponding record in va_attendance gets updated with the new isPresent value. 3. Attendance table in the batch’s roster page re-renders to show the updated value
Add new assignment for a batch	1. User clicks “Add Assignment” button in a batch’s roster page and fills out form with the name of the assignment. 2. addAssignment method in pages/batch/[id].jsx is called which makes a POST API call to api/addassignment with the assignmentName and batchId in the request body. In api/addassignment.js, all students belonging to the batch are fetched and for each of them, a new record is made in va_grades with the new assignment name and default grade of 0. 3. addAssignment methods re-fetches grades data to refresh Grades table with new records. New column appears for grades table with new assignment.
View student enrollment history	1. User clicks on “Enrollment History” button on any row of the table in the Students page. 2. User is redirected to an Enrollment History page for the selected student which is rendered from pages/student/[id].jsx 3. getStudentData method is called which makes a POST API call to api/getstudentdetails with the request body including the studentID. A SQL query is made to fetch batch details of all the batches the student is enrolled in from a JOIN of the vabatches and vastudent_to_batch tables on the batch id. These batches are returned and presented as a table on the Enrollment History page for the student. NOTE: No updates can be made to any of the data presented in the page since it’s meant to be a report of the student’s enrollment history.

RECOMMENDED HOSTING

- Database: DreamHost MySQL DB server. To administer the database (ie: make modifications to tables/schemas/manually manage records) go to https://west1-

phpmyadmin.dreamhost.com/signon.php and enter the login details (shared with Isaac W.). Follow phpMyAdmin documentation fo guidelines on using the portal for database management. The same database can be used for both development and production environments. Use of DreamHost is free and requires proof of 501(c)(3) status.

- Website: Vercel deployment. Log in to the VA team's Vercel account and navigate to the Project page of the visionaid-stats-ng project to view the status of the Production Deployment. Make sure the "STATUS" field is "Ready" and the latest commit message shown under "BRANCH" reflects your understanding of the latest commit of the main branch of the connected github repo. Use of a Hobby account of Vercel is free. There is no foreseeable requirement to upgrade to the Pro account since the primary value of the Pro account is if multiple developers have their own Vercel accounts and want to work together in a Vercel Team as well as performance enhancements and available plugins which are not necessary to run the application.

APPLICATION INSTALLATION

DEV ENV SET-UP

1. Clone the GitHub repository
2. Add all necessary environment variables in the .env file in the project root directory. See the ENVIRONMENT_VARIABLES section below for details.
3. Get the database backup file from /db_backups directory and use it to set up the MySQL database using phpMyAdmin as instructed in the RECOMMENDED HOSTING section.
4. Add yourself as a user with role type MANAGEMENT in the vausers table using your gmail address which will be used for authentication.
5. Open the repo with Visual Studio Code.
6. In the repo, ensure all apiUrlEndpoint constants are set to paths of the form `http://localhost:3000/…'
7. In the terminal from the project root directory, run 'npm run dev'
8. On a web browser navigate to http://localhost:3000

PRODUCTION ENV SET-UP

1. Ensure GitHub repository has Vercel integration specifying the link to the Vercel project in the GitHub project settings.
2. Ensure all apiUrlEndpoint constants are set to paths of the form `https://visionaid-stats-ng.vercel.app/…`
3. Push changes to the "main" branch of the GitHub repo by merging PRs (Pull Requests) made to the repo or pushing changes directly to the "main" branch.
4. Check the deployment status on the Vercel project deployment page. If the deployment fails, logs will be shown on Vercel by clicking on the failed deployment. Follow the guidance in the error messages to resolve errors. Push fixes for the errors and ensure the deployment goes through smoothly.
5. Navigate to the project website https://visionaid-stats-ng.vercel.app to see the newly pushed changes.

ENVIRONMENT VARIABLES

- For local development, in the .env file from the root directory of the project repository, add the following environment variables:
  - MYSQL_HOST: MYSQL DB server hostname (ie: mysql.foo.dreamhosters.com)
  - MYSQL_DATABASE: name of database (visionaid1)
  - MYSQL_USER: database admin username
  - MYSQL_PASSWORD: database admin password
  - JWT_SECRET: generate a JWT token using Postman or command-line JWT token generator
  - GOOGLE_CLIENT_ID from GCP project with OAuth Consent setup
  - GOOGLE_CLIENT_SECRET from GCP project with OAuth Consent setup
- For production deployment on Vercel, ensure that the same environment variables are set in the project Settings.
- Ensure .env file is in the .gitignore file so they are never pushed to the shared repo and always live locally on your machine.

AUTHENTICATION & AUTHORIZATION

Authentication with Google OAuth

1. Navigate to cloud.google.com
2. Menubar dropdown menu on the immediate right of 'Google Cloud'
3. Top right of dialog window: 'NEW PROJECT'
4. Enter 'Project name' and 'Location' if applicable and click 'CREATE' button
5. Click 'Google Cloud' in menubar an <YourProjectName> from the dropdown
6. Click 'DASHBOARD' → 'API APIs & Services'
7. Click 'Credentials' → 'OAuth consent screen' in side menu
8. Select 'External' in the main panel and click 'CREATE'
9. Complete the 'App information' section in the main panel
10. Enter the protected views in the app, e.g., Students, Courses routes and click 'Create'
11. 'OAuth consent screen' in side menu
12. In 'Test users' section of main panel click 'ADD USERS' button
13. Add Gmail addresses for those that will test the app, and save
14. New user(s) now appear in the main panel

In /pages/api/auth/[...nextauth].js, the environment variables GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, and JWT_SECRET are set for the Next.JS next-auth library to connect with the Google OAuth provider set up in earlier steps specified above.

Authorization: RBAC (Role-Based Access Control)

All of the files in the pages directory include a check on the signed-in users' role to ensure they are authorized to view the contents of the page. Some of the pages such as the Batches page have partial redactions depending on the role. For example, the Batch Creation form is not visible or accessible to users with the role ADMINISTRATOR, but the table of batches is visible to all signed-in users but none of the rows are editable or delete-able by users with non-PM or non-MANAGEMENT roles. Here is the full set of authorization rules implemented in the application for Management (MGMT), Program Managers (PM), and Administrators (ADMIN.):

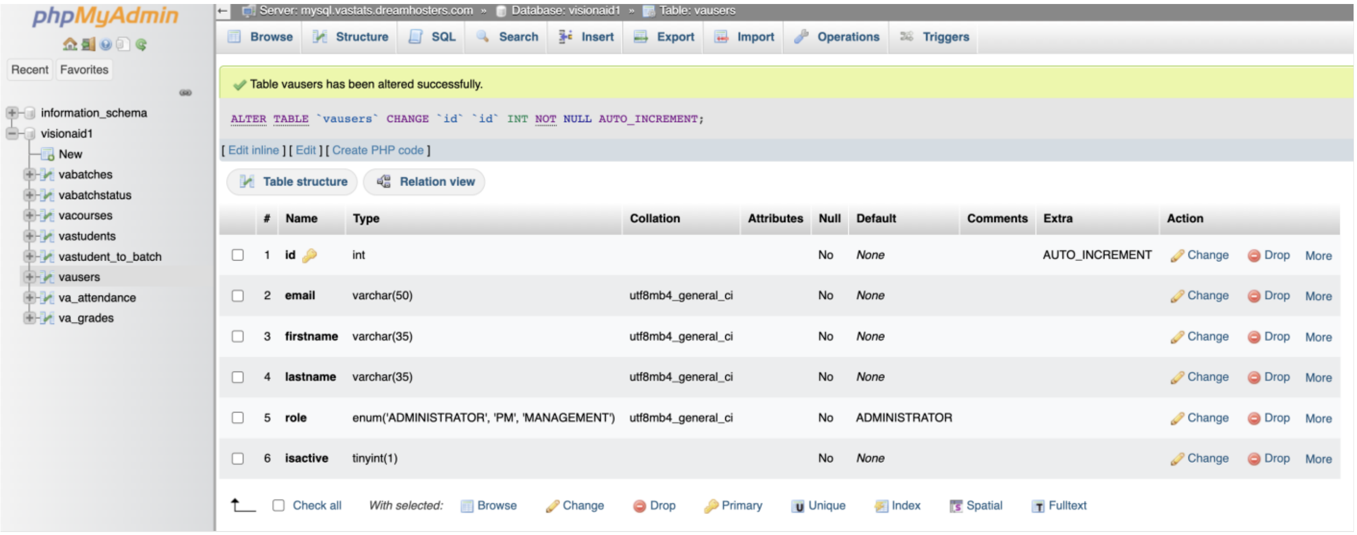
PAGE	RESOURCE	ACTION	MGMT.	PM	ADMIN.	UNAUTH
Home	All	View	🟩	🟩	🟩	🟩
Home	Login Button	Authenticate	🟩	🟩	🟩	🟩
Home	Logout	Authenticate	🟩	🟩	🟩	🟩
Batches	Batch	View	🟩	🟩	🟩	🟥
Batches	Batch	Create	🟩	🟩	🟥	🟥
Batches	Batch	Update	🟩	🟩	🟥	🟥
Batches	Batch	Delete	🟩	🟩	🟥	🟥
Students	Student	Add	🟩	🟩	🟩	🟥
Students	Student	Remove	🟩	🟩	🟩	🟥
Students	Student	Update	🟩	🟩	🟩	🟥
Batch Details	Batch Details	View	🟩	🟩	🟥	🟥
Users	User list	View	🟩	🟥	🟥	🟥
Users	User	Create	🟩	🟥	🟥	🟥
Users	User	Disable	🟩	🟥	🟥	🟥
Batch Details	Batch Roster	View	🟩	🟩	🟥	🟥
Batch Details	Batch Roster	Add student	🟩	🟩	🟥	🟥
Batch Details	Batch Roster	Remove student	🟩	🟩	🟥	🟥
Batch Details	Batch Roster	Update Attendance	🟩	🟩	🟥	🟥
Courses	Course	Create	🟩	🟥	🟥	🟥
Courses	Courses	Update	🟩	🟥	🟥	🟥
Courses	Courses	Delete	🟩	🟥	🟥	🟥
Students	Student	Form	🟩	🟩	🟩	🟩
Batch Details	Batch Roster	Update Grades	🟩	🟩	🟥	🟥

DATABASE BACKUP

- Database backup: provided in the project repo in the /db_backups directory

DATABASE SCHEMAS

The visionaid1 database's table schemas can be viewed from the phpMyAdmin console. For example, to view the vausers table schema, navigate to the vausers table from the side panel and go to the "Structure" tab to view the schema, shown in the screenshot below.



✓ Table vausers has been altered successfully.

```
ALTER TABLE `vausers` CHANGE `id` `id` INT NOT NULL AUTO_INCREMENT;
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**id**	int			No	*None*		AUTO_INCREMENT	Change Drop More
□	2	**email**	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
□	3	**firstname**	varchar(35)	utf8mb4_general_ci		No	*None*			Change Drop More
□	4	**lastname**	varchar(35)	utf8mb4_general_ci		No	*None*			Change Drop More
□	5	**role**	enum('ADMINISTRATOR', 'PM', 'MANAGEMENT')	utf8mb4_general_ci		No	ADMINISTRATOR			Change Drop More
□	6	**isactive**	tinyint(1)			No	*None*			Change Drop More

TABLE COMPONENT OVERVIEW

We use the Table component (/components/Tables.jsx) to display all tables on the website, such as the table of batches, students, users, and attendance/grades tables for batch rosters.

To utilize the table component, instantiate it with the constructor such as

```
<Table columns={coursesColumn} tableData={dataResponse} isDelete={isDelete} onDeleteClick={handleDeleteCourse}
isEditable={isEditable} onEditSave={handleUpdateCourse} Title={'Courses List'} />
```

Ensure the column headers are defined for the columns attribute, specifying at least the name (displayed as column header) and accessor (column name in dataset returned) for each column. Other properties for a column header element includes:

- column.width, which is the width of the column displayed on the webpage as a percentage of the total width
- column.type which is the type for the values in that column. For example, if the type is `enum` column.type should be set to 'enum'.
- column.availableValues is an array of possible values if the column.type is 'enum'.

tableData is the dataset to be displayed as a table on the webpage.

isDelete and isEditable should be set to true if the rows of the table should be editable and deletable. If they are, the onDeleteClick and onEditSave functions should be defined and specified as input for the Table.

Table component's helper functions are defined in the /utils/tableHelper.js file.

PWA (Progressive Web Application)

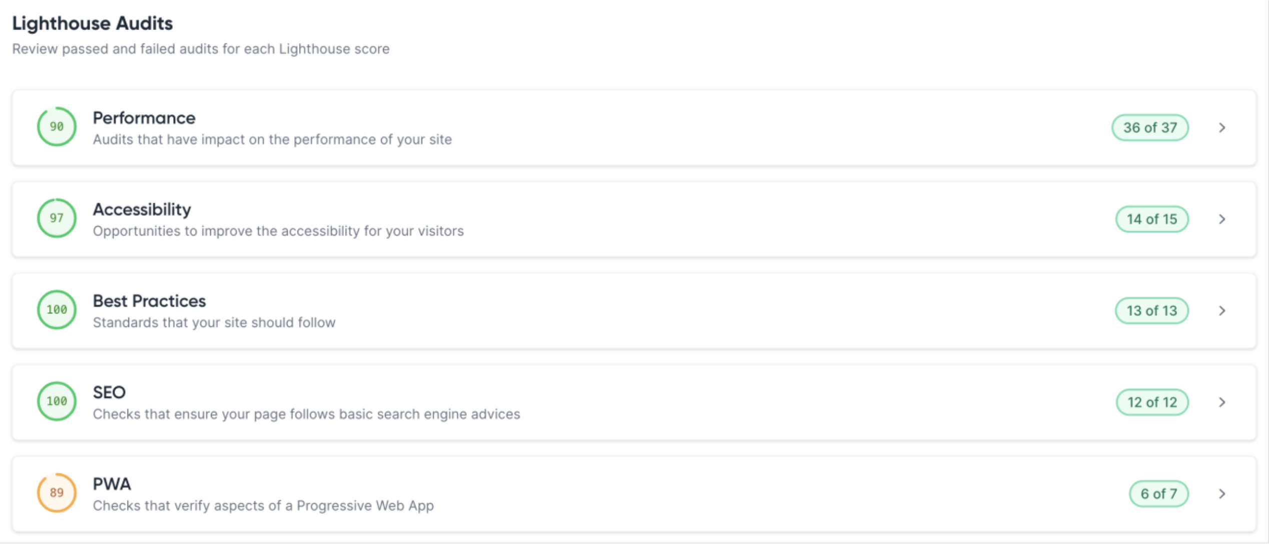
This project leverages PWA to make the application mobile-friendly. PWA is enabled on every website page by including a Next.JS <Head/> component to each page with properties essential for enabling PWA. If adding new pages to the project, add the following <Head/> component to the HTML:

```
<Head>
  <title>VisionAid</title>
  <meta
    name='description'
    content='A nonprofit, advocating on behalf of persons with vision issues of any type' />
  <meta name='theme-color' content='#ffffff' />
  <link rel='icon' href='/favicon.ico' />
  <link rel='apple-touch-icon' href='/apple-touch-icon.png' />
  <link rel='manifest' href='/manifest.json' />
  <link rel='preconnect' href='https://fonts.gstatic.com’ crossOrigin />
  <link rel='preload' as='style' href='https://fonts.googleapis.com/css2?family=Work+Sans:wght@400;500;700&display=swap' />
  <link rel='stylesheet' href='https://fonts.googleapis.com/css2?family=Work+Sans:wght@400;500;700&display=swap' media='print' onLoad="this.media='all'" />

  <noscript>
    <link rel='stylesheet' href='https://fonts.googleapis.com/css2?family=Work+Sans:wght@400;500;700&display=swap' />
  </noscript>
</Head>
```

DELIVERABLE SCREENSHOTS

Lighthouse metrics:



Lighthouse Audits

Review passed and failed audits for each Lighthouse score

90 **Performance** — Audits that have impact on the performance of your site — 36 of 37 ›

97 **Accessibility** — Opportunities to improve the accessibility for your visitors — 14 of 15 ›

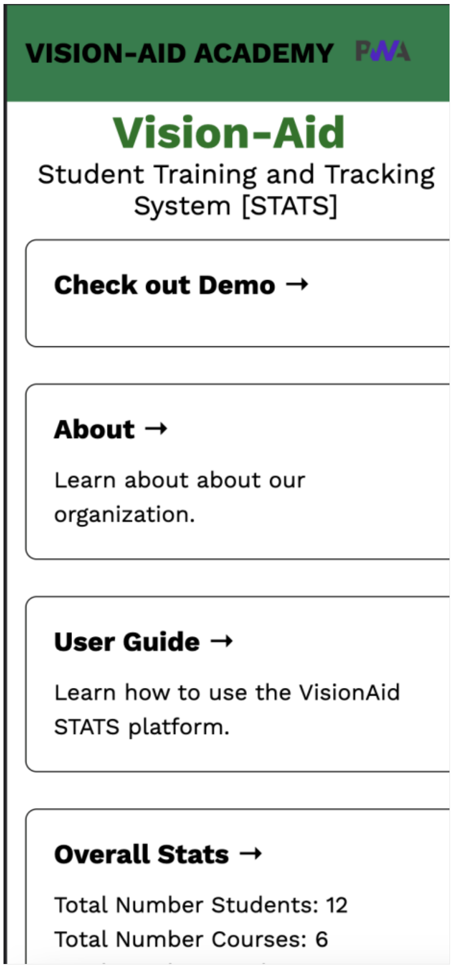
100 **Best Practices** — Standards that your site should follow — 13 of 13 ›

100 **SEO** — Checks that ensure your page follows basic search engine advices — 12 of 12 ›

89 **PWA** — Checks that verify aspects of a Progressive Web App — 6 of 7 ›

Emulator (iOS):

1. Portrait



Vision-Aid

Student Training and Tracking System [STATS]

Check out Demo →

About →

Learn about about our organization.

User Guide →

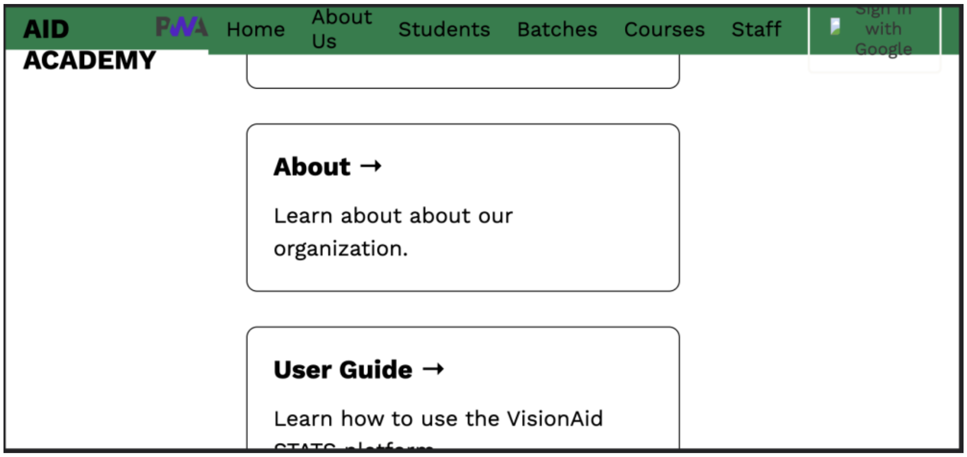
Learn how to use the VisionAid STATS platform.

Overall Stats →

Total Number Students: 12

Total Number Courses: 6

2. Landscape



About →

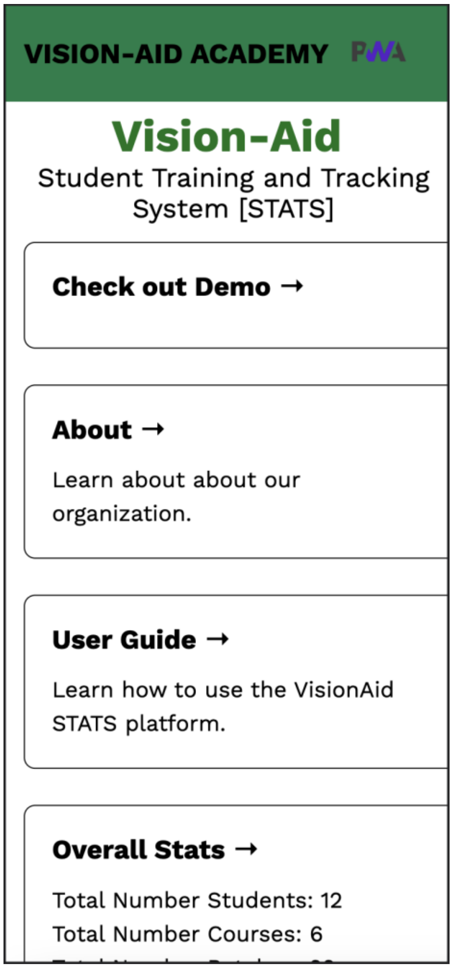
Learn about about our organization.

User Guide →

Learn how to use the VisionAid STATS platform.

Emulator (Android):

1. Portrait



Vision-Aid

Student Training and Tracking System [STATS]

Check out Demo →

About →

Learn about about our organization.

User Guide →

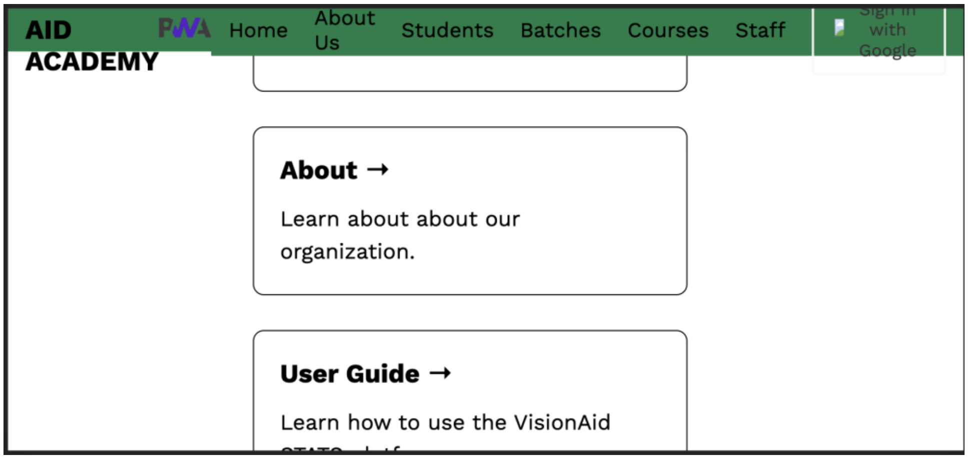
Learn how to use the VisionAid STATS platform.

Overall Stats →

Total Number Students: 12

Total Number Courses: 6

2. Landscape



About →

Learn about about our organization.

User Guide →

Learn how to use the VisionAid

KNOWN ISSUES

- Performance issue: Enrolling a student to a batch is functional, but takes a long time since a new database record has to be created in va_attendance for each student and each date of the batch. For example, if a batch runs for 3 months and there are 3 sessions per week, assuming there are 4 weeks per month, 3*3*4=36 new records need to be made in the va_attendance table per student. Additionally, if there are 10 assignments already defined for the batch, 10 new records will be created for each student in the va_grades table. Since enrolling a student is a batch operation that includes adding all these data records, it takes an average of 1.5 minutes for the operation to complete today. Improvement need to be made to make this more performant.

- There aren't many guardrails for user inputs in forms. Input checks should be added to ensure consistency in data formats.

PENDING FEATURE REQUESTS

- Enable Microsoft Outlook authentication. Vision Aid staff primarily uses Microsoft Outlook rather than Gmail.
- Batch Roster page: Present Unassigned Students in a table format, just like Students page table, where multiple student records of unassigned students can be selected at a time to add to a batch.
- Students page: show 25 student records at a time and have prev/next buttons to navigate.
- Additional reports on how many students total are enrolled in each course, etc.
- Automatic final grade calculation

PARTNER STATEMENTS

PENDING – week of 5/1	Partner understands developer documentation
DONE with Ruthvik – 4/27	Partner has gone through an installation walk-through