Competition-Hub Design Document

The front-end of the website will be developed using a combination of HTML, CSS and JavaScript. The bootstrap framework, which features predefined CSS, will be used to speed up development.

The back-end will be developed on the NodeJS, which allows one to use JavaScript on the server side. Using NodeJS is based on installing the appropriate packages using the Node Package Manager, otherwise known as npm. The most relevant npm packages used and their functions are as follows. Note that other packages were also installed to complement these.

Name	Function
Express	The framework on which the website will be built on. It is the most documented, supported and used framework, and thus I felt it would be apt for the task.
Passport	A package that allows for professional level authentication using cookies and hash encryption. It will be used to create the login and user functionality.
Mongoose	This package is the interface with the MongoDB database. All the adding, deleting and editing of records will be performed using this package.
EmailJS	A package that allows the sending of emails. It will be used to implement the notification email system.
Excel4Node	A package that allows the creation of excel files. It will be used to implement the registrant list download feature.
EJS	It is a package that serves as a templating language. Its function is to allow for the effective display of information from the backend onto the front end.

This design document will detail the route structure, what routes are accessible from where, a brief overview on the actions that take place in each route and a description of the database system including what the data members are composed of and what actions are performed on these members. I will start with the description of the database system.

I designed my application so that I would only need two collections of objects in the database. One to keep track of the users and one to keep track of the competitions. Before explaining the structure of each of these objects, I will detail the relevant Data Types that the Database system I have employed, called MongoDB, uses. In this system, an entry in a collection is called a *document.* The types used are as follows:

Type Name	Type Description
String	A sequence of alphanumeric characters
Number	Stores numeric data
Boolean	Stores a value of either True or False
Array	Stores a list of items of a particular type.
Object	Stores data in the form of a JSON object
ObjectID	Stores a unique key id for an individual entry.

A collection is created using the concept of a model. These models are essentially JavaScript code that defines the properties that a document has and their respective types. An important point to note about the nature of this system is that not all entries need to contain all of these properties. This feature is taken advantage of in the User collection when distinguishing between a normal user and an admin user.

The Model for a document in the Competitions collection, along with a description of what each property is used for and example data is as follows:

Name of Property	Type of Data Stored	Description	Example Data
Name	String	The name of a competition.	1. “FLL” 2. “IHMC”
Category	String	A broad category into which the competition can be put into.	1. “Robotics” 2. “Music”
Grades	String	The grades to which the competition is open to.	1. “9,10,11,12” 2. “IB1,IB2”
Houses	String	The houses for which the competition is open to.	1. “Cygnus,Aquila” 2. “Orion”
Description	String	A description of what the competition entails, also includes.	1. “A competition in which one has to debate.”
Registrations	Array of type Object	A list of the registrations that	Check note.

		the competition has.	
Posted By	String	Stores the Email ID of the admin that posted the competition	“pallipadanjoseph@gmail.com”
Posted On	String	Stores the date and time at which the competition was posted.	“2018/09/21”
Registration Open	Boolean	Data that determines if students can register for the competition or not.	True, False
ID	ObjectID	The unique Id of the competition.	ObjectId(“5ba208bb770a1708922b5b0c”)

Notes:

An example data for the Registrations is as follows:

[

{

name: 'Nimy Baby',

grade: 'IB2',

username: 'nimybaby@yahoo.com',

details: 'Nothing asked.',

time: 1537357395076

}],

The idea of having another collection to represent registrations was considered but ultimately, it was decided that the extra structure it brought was not worth it compared to the amount of extra code that would need to be written to make it happen.

The choice of representing the *Grades* and *Houses* properties as Strings rather than Arrays might seem counter-intuitive at first but as with the registrations, the extra code required to convert the data from the form into an Array and other logistical issues led me to choose to just represent them as Strings separated by commas.

When understanding how the documents in this collection are used, the aforementioned point about all documents not needing to have all properties is crucial. When I was designing the Student and Admin functionality of the application, I had initially intended to have separate collections for Students and Admins. However, this caused problems with the node module I was using for the authentication functionality. Thus, I decided to circumvent the issue by creating a single User model which combined the properties of both the Admin and Student users along with a Boolean variable to distinguish between the two.

That said, the model for a document in the User collection, along with a description of what the property is used, some example data and whether the property is used by an Admin, Student or both is as follows:

Name of Property	**Type of Data Stored**	**Description**	**Example Data**	**Used By**
Username	String	The Email Id of the User	“pallipadanjoseph@gmail.com”	Both
Salt	String	A String that can be used to decode the hashed version of the password	"89466c16519a354db5dd2c8 41a2a08110d302d51a4b545 68ced8b71a629c33c8"	Both
Hash	String	The Hashed version of the password	Too long to provide an example. But it is essentially a string that looks like the salt but is approximately 10 times longer.	Both
Name	String	The name of the user	“Joseph Pallipadan”	Both
Grade	String	The grade of the user	1. “IB2” 2. “11”	Student
House	String	The house of the user	“Cygnus”	Student
isAdmin	Boolean	Whether or not the user is an admin	True	Both
Competitions	Array of Type ObjectId	An array which holds references to all the competitions a particular user who is an admin has posted	[ObjectId("5ba0ac35a649190d43ebcaa6"), ObjectId("5ba208bb770a1708922b5b0c")]	Admin

To understand the remaining portion of this document, it is important to understand the concept of route paths in Express.Js. A route path is used to define an endpoint where requests can be made. It does so with the combination of a request method. In Express, route paths can be string patterns or regular expressions. My website works by sending requests to specific routes based on what action needs to be taken. Before explaining the routes, I will explain the four types of requests that my website can make to a route.

GET

This type of request is used mainly to obtain some information from the server. In my application I use it to send the HTML code for the front end of the different pages.

POST

This type of request is used when information needs to be sent back to the server. For example, from a form the user fills in.

PUT

This request is used when an item from a database needs to be edited.

DELETE

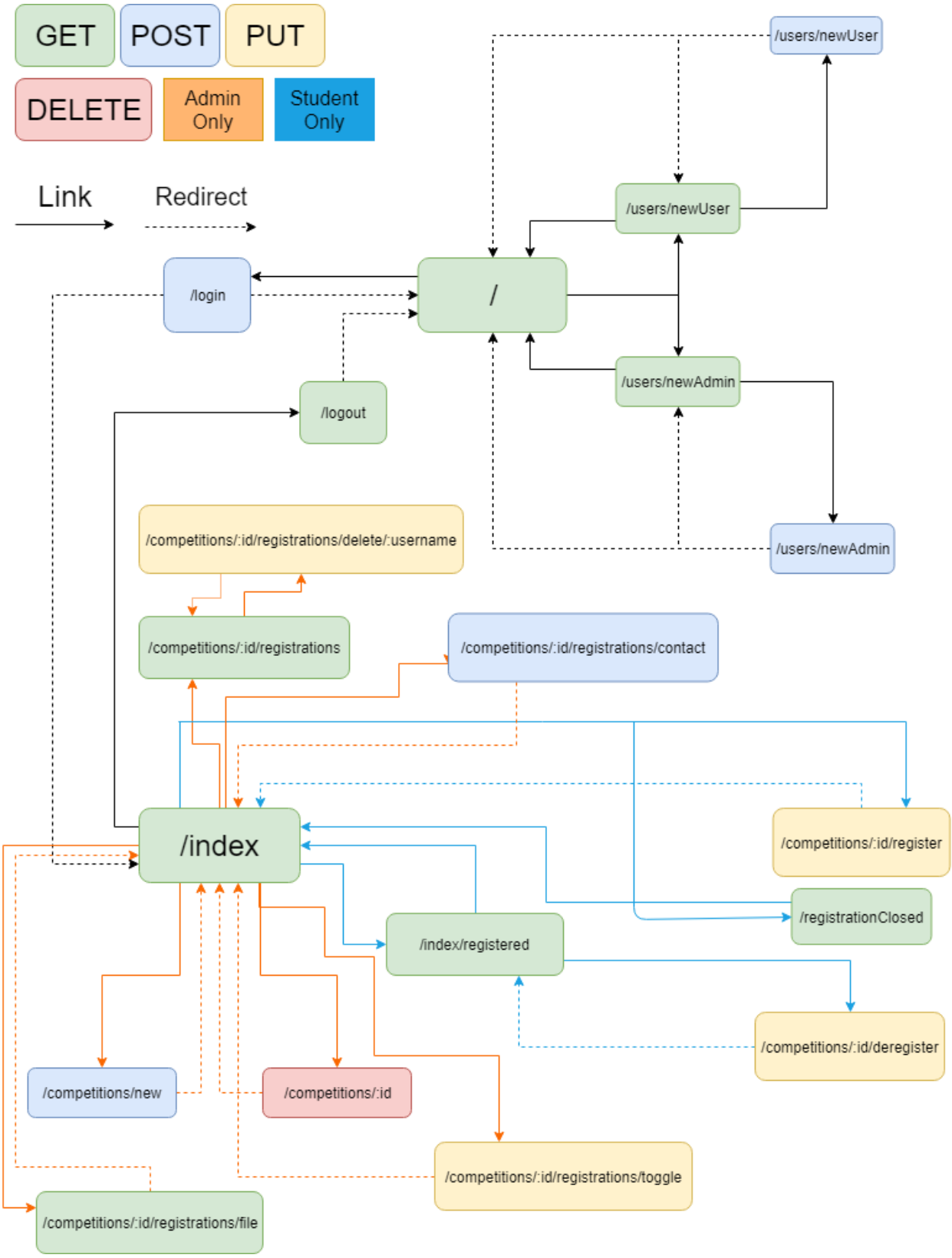
This type of request is used when an item from the database needs to be deleted.

It is important to note here that these request types are not necessarily binding. A GET request can technically be used to delete an item if the appropriate information is there in the route. However, I have refrained from doing that as it breaks the conventions defined by the RESTful routes, a set of guidelines which all Express applications should ideally follow.

With that said, the table that follows shows all the routes that I have defined in my application, along with type of request they handle and a brief description of what happens in each.

Name	Type	Description
/	GET	Displays Landing Page
/users/newUser	GET	Displays Student Sign Up Form
/users/newAdmin	GET	Displays Admin Sign Up Form
/users/newUser	POST	Registers User In Database
/users/newAdmin	POST	Registers Admin In Database
/login	POST	Logs in a user
/logout	GET	Logs out a user
/index	GET	Displays added competitions for admin or available competitions for student
/competitions/new	POST	Create new competition
/competitions/:id	DELETE	Delete the competition with the id in the path
/index/registered	GET	Display competitions that a user has registered for
/registrationClosed	GET	An edge case for if an admin closes the registration for a competition in the time period between the user opening his index and registering.
/competitions/:id/register	PUT	Registers a user for the competition with the id in the path
/competitions/:id/deregister	PUT	Deregisters a user from the competition with the id in the path
/competitions/:id/registrations	GET	Displays all registrations for the competition with the id in the path
/competitions/:id/registrations/toggle	PUT	Toggles the registration open or closed status for the competition with id in the path
/competitions/:id/registrations/contact	POST	Sends an email containing a message from the adminto all the registrants of the competition with the id in the path.
/competitions/:id/registrations/delete/:username	PUT	Deletes the user with the username in the path's registration from the competition with the id in the path.
/competitions/:id/registrations/file	GET	Creates an excel file containing all the details for registrations of the competition whose id is in the path.

The diagram on the next page shows the flow of my website.


GET
POST
PUT
DELETE
Admin Only
Student Only
Link
Redirect
/users/newUser
/users/newUser
/login
/
/users/newAdmin
/logout
/competitions/:id/registrations/delete/:username
/users/newAdmin
/competitions/:id/registrations
/competitions/:id/registrations/contact
/index
/competitions/:id/register
/registrationClosed
/index/registered
/competitions/:id/deregister
/competitions/new
/competitions/:id
/competitions/:id/registrations/toggle
/competitions/:id/registrations/file

Test Plan

Note that in all cases where the user needs to enter data, checks have been implemented on the front end to ensure that parameters are not missed. These do not need to be formally tested and thus checks with regards to missing data were omitted from the testing process.

Also, in any place where 'available competitions' or similar terminology is used, it refers to sample data added.

No.	Action To Be Tested	Testing Process
1	Loading Landing Page	1. Visit website URL on mobile, tablet and PC. 2. Ensure that login form and links for signing up load.
2	Loading User Sign Up Form	1. Click on link to user signup form. 2. Ensure that the form loads (name, password and email textboxes plus dropdowns for house and grade) along with the back button and submit button.
3	Duplicate Emails Blocked from Signing Up (Same for user and admin)	1. Create an account with an email. 2. Attempt to create another one with the same email. 3. Ensure that the page reloads with a message indicating that a user with that email already exists.
4	Loading Admin Sign Up Form	1. Click on link to user signup form. 2. Ensure that the form loads (name, password, email and admin code textboxes) along with the back button and submit button.
5	Incorrect Admin Code Blocked from Signing Up	1. Attempt to create an admin account with the wrong admin code. 2. Ensure that the page is reloaded with a message indicating that the admin code was incorrect.
6	Signing Up as User	1. Enter sample details on the user signup form and sign up. 2. Ensure that the user is redirected to his index page.
7	Signing Up as Admin	1. Enter sample details on the user signup form and sign up. 2. Ensure that the admin is redirected to his index page.
8	Logging In (Same for user and admin)	1. Enter details of an account previously created and click the login button. 2. Ensure that a redirect to the index route occurs.
9	Ensuring Incorrect Logins Are Handled	1. Attempt to login with incorrect details (a non-existing account or an incorrect password/email) 2. Ensure that the page is reloaded with a message indicating that the details were incorrect.
10	Logging Out (Same for User and Admin)	1. Click on the logout button on the index route. 2. Ensure that a redirect back to the landing page occurs.

11	Loading Index Page for Users	1. Log in as a user. 2. Ensure the index page loads, indicates who is logged in, and shows buttons to logout, switch between registered and non-registered competitions and filter competitions by name and category along with a list of all the competitions that the user can register for, with a button each to register. 3. When one of the register buttons is clicked, a modal should pop up, containing a text box to enter registration details.
12	Loading Index Page for Admins	1. Log in as an admin. 2. Ensure the index page loads, indicates who is logged in and shows buttons to logout and add competitions. 3. When the add-competitions button is clicked, a modal should pop up containing the form to add competitions should appear. 4. All the competitions the admin has added should be listed along with all their properties, with buttons to toggle registration status, contact all participants, download the registrant list, and delete the competition.
13	Adding New Competition	1. Log in as admin. 2. Add a competition using sample data. 3. Log in with different user accounts and ensure that the competition appears to only users whose properties match that of the competition (grade and house).
14	Deleting Competition	1. Log in as an admin. 2. Delete one competition from the list using the delete button. 3. Ensure that it does not appear on the admins list after the page reloads. 4. Log in with different user accounts and make sure the competition does not appear on any of the registered competition or non-registered competition lists.
15	Loading a User's Registered Competitions Page	1. Log in as a User and click on the registered competitions link. 2. Ensure that the page loads and contains a list of all the competitions that the user has registered for, along with buttons to deregister from each.
16	Ensure Case When User Registers After Registration Closing Is Handled	1. Log in as a user such that a competition 'x' appears on the list to register. 2. Before the user clicks a button, on another device, login as an admin and close the registration for the same competition.

		3. Attempt to register for that competition as the user. 4. Ensure that a redirect to a page indicating that the registration was closed in between the time the user loaded the page and registered for the competition.
17	Registering for A Competition	1. Log in as a user and click on the register button for an available competition. 2. Enter sample data in the modal that appears and submit. 3. Ensure that this competition only appears in the registered competitions section of the user's accounts. 4. Ensure that the admin of the competition of the competition receives an email notifying them of the registration. 5. Ensure that the registration appears with all its details in the registrations page of the competition.
18	Deregistering from A Competition	This is the exact opposite of the registrations action except step 2 is not included.
19	Loading Registrations Page for A Competition	1. Log in as an Admin. 2. Click on the view registrations button for any of the available competitions. 3. Ensure that a page containing all the registrations for said competition loads, along with links to go back and buttons to delete each registration.
20	Toggling Registration for A Competition	1. Log in as an Admin. 2. Toggle the registration of an available competition using the open/close registration button. 3. Depending on to what the competition was toggled, ensure that the competition appears on or disappears from the lists of users who are eligible to register.
21	Delete A Registration from a Competition	1. Log in as an Admin. 2. Click on the view registrations button for any of the available competitions. 3. Click on the delete registration button for any one of the competitions. 4. Ensure that an email is sent to the registrant indicating that his registration has been deleted and also ensure the registration no longer appears in the registrations of the competition.
22	Obtaining Excel File Containing	1. Log in as an Admin.

	Details Regarding All Registrations of A Competition	2. Click on the download registrations as a file button for any of the available competitions. 3. Ensure that an excel file is downloaded and that it contains all the registrations along with their details of the competition.
23.	Search by Keyword	1. Log in as user ensuring that there are competitions which he can register from appearing in his index. 2. Search for name of a particular competition and ensure that only it appears.
24.	Search by Category	1. Log in as user ensuring that there are competitions which he can register from appearing in his index. 2. Filter by a particular category and ensure that only competitions of that category appear.