Streamlining the QUEST Honors Programs Internal Systems (V 1.0.0)

CMSC435: Software Engineering

Presented by:

Samuel Baer (sambaer7@gmail.com)

Jacqueline Deprey (jacqueline.deprey@gmail.com)

Sofia Gonzalez (sofiegonzalez98@gmail.com)

Mark Kapuscinski (kapski@terpmail.umd.edu)

William Pierson (wpierson@terpmail.umd.edu)

Luke Vacek (lvacek@terpmail.umd.edu)

Ronaldo Zavala (ronyzav15@gmail.com)

TABLE OF CONTENTS

1. Introduction

As the QUEST Honors Program has grown over the past 27 years, the program directors have off-and-on built up their technology infrastructure to handle the program's pressing needs at any given time. As a result, the program has several different systems in place that process their own copies of the same data. Because managing the different systems is time consuming and confusing at times, QUEST initiated this project with the QUBIT CMSC435 Team to explore how they can simplify their process workflow while still maintaining all their existing functionality.

The objective of this project is to create a new unified management system and provide a dashboard to display projects and allow administrators to manipulate data. To accomplish this, the QUBIT team has decided to create a unified database to house the current QUEST data, an API to facilitate future development, a web application to display projects created by QUEST students, and thorough documentation for future developers. This unified database and API will set the groundwork in place to allow the flow of data between the many disparate applications QUEST admins currently interact with. This will allow for the time that was used transferring data to be saved. Due to the fact the development effort to interface the existing systems with the new API exceeds the team's time, after meetings with Jess and Danny the decision has been made to instead build a Project Dashboard, adding additional tools to enhance the QUEST admin workflow. The goal is to provide a functional system that makes the QUEST workflow more efficient while being flexible enough to be extended as QUEST grows.

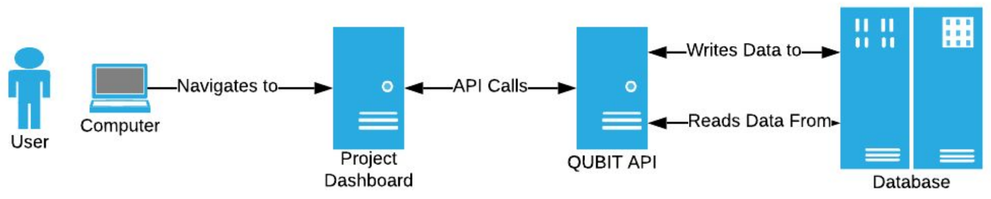


Fig 1. Relational diagram depicting how platforms and users are connected

This system works by having any of QUEST's stakeholders navigate to a singular Project Dashboard web page (https://projects.questumd.com/) which displays the work of QUEST students. Students, faculty and project champions are then able to login and manipulate any necessary data via calls to the API. The API will then write and read data from an uniform database containing information on all QUESTees, projects, teams, faculty, etc. These three key components work hand in hand to simplify the Quality Guild's existing administrative workflow.

2. Database

The database serves as a single source of truth for all future QUEST applications. Tables and relationships are organized in an object-oriented fashion. Since the data is highly relational and class based, an object-oriented model was selected over a hierarchical model. The slight query overhead caused by more distinct tables is not significant given the low frequency of queries. Additionally, the object-oriented model allows for increased clarity and modularity, allowing for easier continued development. In order to store files, such as project poster PDF files, the team uses an AWS S3 bucket as additional file storage, allowing for links to be stored in the database.

The relational database is built using a SQL based server and hosted on the preconfigured QUEST AWS RDS instance, provided by our client.. In order to store files, such as project PDFs, an additional S3 bucket is used that can be referenced from that database as additional file storage. This hosting is in a way that is consistent with current Quest Applications.

The specific platform has been narrowed down to be MySQL. MySQL was chosen because it was what the prior QUEST systems were written with as well as being the tool the team has the most experience with. The team plans on using migrations to build the database directly from the schemas developed.

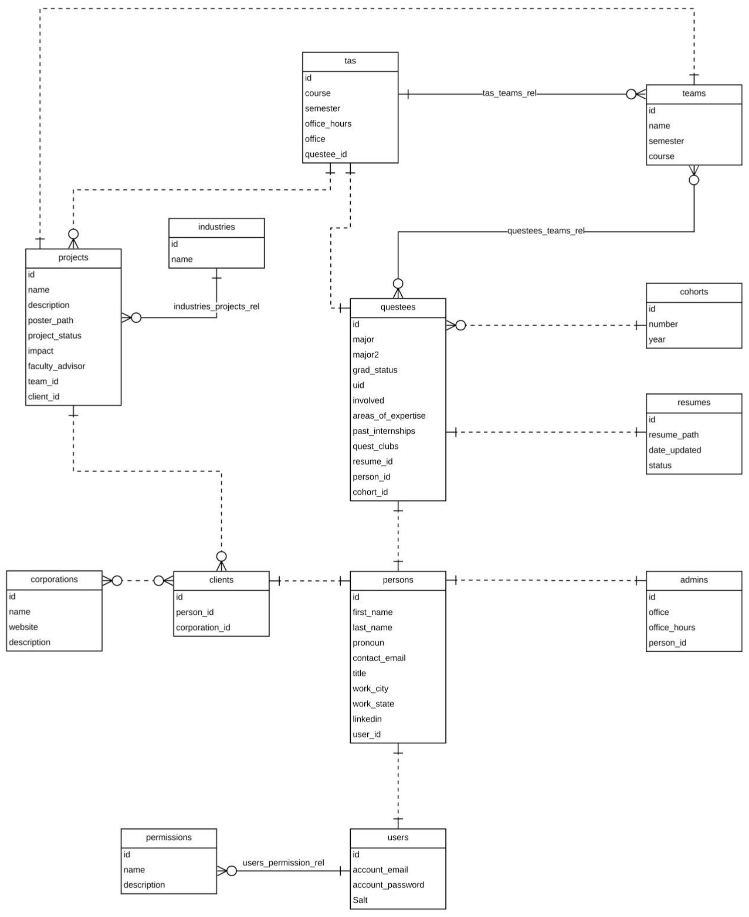


Fig 2. Entity Relationship Diagram of Database

2.1 Users

This table contains the account information of all users/stakeholders of QUEST.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every account created within the system
account_email	String	The email the account is associated with
account_password	String	The encrypted password the account is associated with
salt	String	The salt used to hash the password for security purposes

2.2 Persons

All users are persons and have information in a respective table describing all of their personal attributes. This information is separated out to make it easier to find and to prevent the need of administrators to edit the data in the User table after account creation.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every person created within the system
first_name	String	The preferred first name of the person
last_name	String	The legal last name of the person
contact_email	String	The preferred email of the person for all QUEST communication
pronoun	String	The pronouns the person uses/identifies with

title	String	The official job title of the person
work_city	String	The city in which the person works
work_state	String	The state (if within the US) or country (if abroad) where the person works
linkedin	String	The linkedin url for the student
user_id	String	The foreign key identifier linking the person to the account created in the User table

2.3 Admins

Administrators are persons with unique properties not shared by other stakeholders. This table stores just that and is linked to the Persons table via the person_id.

Attribute	**Type**	**Description**
id	Integer	The primary key identifier unique to every administrator created within the system
office	String	The location of the office of administrator written as "(UMD Building Abbreviation)(Room Number)" ex. "VMH1407D"
office_hours	String	The hours in which the administrator will be in their office with their door open for students to stop by
person_id	String	The foreign key identifier linking the admin to the person created in the User table

2.4 Corporations

Administrators are persons with unique properties not shared by other stakeholders. This table stores just that and is linked to the Persons table via the person_id.

Attribute	**Type**	**Description**
id	Integer	The primary key identifier unique to every corporation created within the system
name	String	The unabbreviated name of the corporation
website	String	The official website of the corporation
description	String	A general description of the company, including what the company does, when it was founded, where its headquarters is located, the industry it is in.

2.5 Clients

Clients are the person - Project Champions of QUEST projects. Clients serve as a representative of the company they work for to QUEST and this table contains the necessary links to reflect just that.

Attribute	**Type**	**Description**
id	Integer	The primary key identifier unique to every client created within the system
person_id	Integer	The foreign key identifier linking the client to the person created in the User table
corporation_id	Integer	The foreign key identifier linking the client to the information on the company they work for located in the

		Corporation table

2.6 Teams

Teams are a core part of QUEST's curriculum, so this table allows the information for them to be stored as a unit.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every team created within the system
name	String	The team name created by the students on it
semester	String	The semester in which the team was founded
course	String	The QUEST curriculum course that the team was created in

2.7 Projects

Projects are created by teams within a specific class with QUEST.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every project created within the system
name	String	The title of the project given by the students
description	String	The description of the project includes an overview of the scope of the project and the team's approach to solving it. This description should be no more than 200 words
course	String	The QUEST curriculum

		course that this project was created for
poster_path	String	The file path to the PDF poster associated with the poster
project_status	Integer	The status of the project either submitted, approved or rejected
impact	String	A written description of the impact the project has. This field should include as many quantifies as possible while still abiding by the signed NDA
faculty_advisor	String	The faculty advisor that assisted the team through the duration of the project. For 190H teams, this should be replaced with the team mentor’s name instead. If the project did not have a faculty advisor such as in 390H, the course professor’s name should be listed instead
team_id	Integer	The foreign key identifier linking the project to the team that worked on the project located in the Teams table
client_id	Integer	The foreign key identifier linking the project to the Project Champion that led the project located in the Client table

2.8 Cohorts

Cohorts are identifiers for QUESTees that consist of the number they are known by as well as the year they are in at the University of Maryland.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every cohort created within the system
number	Integer	The number of the cohort within the QUEST program
year	Integer	The year in school students in the cohort are in. This is used for tracking what QUEST course the student should be enrolled in at any given time

2.9 Resumes

Students have the potential to upload their resume to the database to facilitate with the creation of the biannual-QUEST resume database.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every resume created within the system
resume_path	String	The file path to where the resume is stored.
date_updated	String	The date in which the resume was uploaded
status	Boolean	A bit value representing whether or not the student wants their resume to be included in the QUEST resume database

2.10 QUESTees

QUESTees are students in the QUEST Program that have all the attributes of the person entities as well as a cohort associated with them. As they work their way through the curriculum, they are members of a variety of teams and complete several projects thereon.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every QUESTee created within the system
major	String	The primary major of the QUESTee as it appears on their university transcript.
major2	String	The secondary major of the QUESTee as it appears on their university transcript.
grad_status	Integer	The graduation status of the QUESTee
involved	Integer	A value indicating whether or not a QUESTee wants to stay involved with QUEST upon graduating
areas_of_expertise	String	A description of what the QUESTee's area of expertise are to aid with QDes and assisting other students
quest_clubs	String	A description of the QUEST clubs students are involved with
resume_id	Integer	The foreign key identifier linking the QUESTee to their uploaded resume located in the Resume table
person_id	Integer	The foreign key identifier linking the QUESTee to their person demographic information located in the Person table

cohort_id	Integer	The foreign key identifier linking the QUESTee to their cohort located in the Cohort table. This will be 0 if the student was not an official member of QUEST

2.11 TAs

Teaching Assistants (TAs) are QUESTees who assist Administrations with the facilitation of courses in the QUEST curriculum.

Attribute	**Type**	**Description**
id	Integer	The primary key identifier unique to every Teaching Assistant created within the system
course	String	The course that the QUESTee is a teaching assistant for
semester	String	The semester in which the QUESTee is Teaching Assistant
office	String	The location of the office or space where a Teaching Assistant will be available to students written as “(UMD Building Abbreviation) (Room Number)” ex. “VMH1407D”
office_hours	String	The hours in which the Teaching Assistant will be available to students to stop by
questee_id	Integer	The foreign key identifier linking the Teaching Assistant to their QUESTee personal information located in the QUESTee table

2.12 Industries

Industries are ways to categorize corporations to facilitate the scoping process for other potential projects.

Attribute	Type	Description
id	Integer	The primary key identifier unique to every industry category created within the system
name	String	The broad description name for the industry that the company services

2.13 Industries_Projects_Rel

The purpose of this table is to link the industry in which a project deals with to the project itself. This is represented in the ERD as a solid line with a label attached.

Attribute	Type	Description
industry_id	Integer	The foreign key identifier representing the industry the project deals with
project_id	Integer	The foreign key identifier representing the project

2.14 QUESTees_Teams_Rel

The purpose of this table is to link a QUESTee with the teams that they are on. This is represented in the ERD as a solid line with a label attached.

Attribute	Type	Description
questee_id	Integer	The foreign key identifier representing the QUESTee
team_id	Integer	The foreign key identifier representing the team

2.15 Users_Permissions_Rel

The purpose of this table is to link a User with the permissions associated with their account type. This is represented in the ERD as a solid line with a label attached.

Attribute	**Type**	**Description**
user_id	Integer	The foreign key identifier representing the user account
permission_id	Integer	The foreign key identifier representing the permissions the account has

2.16 TAs_Teams_Rel

The purpose of this table is to link a Teaching Assistant with the teams in the class that they assist with. This is represented in the ERD as a solid line with a label attached.

Attribute	**Type**	**Description**
ta_id	Integer	The foreign key identifier representing the Teaching Assistant
team_id	Integer	The foreign key identifier representing the team in the class the user is in.

2.17 Security

While at the moment anyone is able to create an account, we intend for this functionality to be removed in the future and only allow the Quality Guild to have the ability to create new accounts for incoming QUESTees and Project Champions. We use an https encryption scheme to passwords to the API. Inside the database itself, passwords will be encoded with a hashing algorithm with a salt.

2.18 Testing

The acceptance tests outlined in the Green Light document were carried out on the databases created as a result of this project. While the exact design of the database has changed, these tests still apply. For each of the testing procedures, API calls were used to interact with the database. Each test consisted of a series of calls containing JSON bodies that were used to fill in the database, these commands can be seen in the “test_commands.txt” file in qubit/qubitapi repository. A full outline of the tests regarding the database and the sign offs there on can be found in the qubit/database folder in the repository.

3. API

The Application Program Interface (API) is what is used to communicate with the database. Calls to the API read and write to the database, and serve as the only means of manipulating this data. The API is implemented with Flask, a Python web framework. Using Flask allows individuals who develop further QUEST systems to be consistent since Python is used in other QUEST backend systems. The API also uses resource routing to quickly declare all of the common routes for a given resourceful controller.

The API is designed with the database structure in mind, allowing for direct access to various tables depending on user permissions. The API covers standard GET, PUT, POST, and DELETE operations for all tables within the database with a few exceptions. The API also enables search queries. This provides a strong backend for the project dashboard and future QUEST developers by allowing creative and specific queries. Here are the specifics:

3.1 Users

Users represent members of QUEST that must have an account in order to store their QUEST information. Every user type as described in user characteristics is a user with the exception of the public. All users store an email address, username, password, and a set of permissions.

(POST) **createUser**(String email, String username, String password, String permission_name, String permission_description)

>Creates a user in the database
>Creates their default or selected permissions
>Returns result indicating success or failure

(PUT) **updateUser**(int id, String email, String username, String password)

>Updates selected user (based on id)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getUser**(int id)

>Gets selected user (based on id)
>Returns user information on success
>Returns indication of failure on fail

(DELETE) **deleteUser**(int id)

>Deletes selected user (based on id)
>Returns result indicating success or failure

(GET) **getAllUsers**()

>Returns all users information on success
>Returns indication of failure on fail

(GET) **searchUsers**(int id, String email, String username, String password)

>Returns all user information that matches all non-empty search criteria (parameters) on success

>Returns indication of failure on fail

3.2 Permissions

Permissions enable developers to limit how much access a given user has to information within the database. As such, it is vital that permissions for each user must be easy to access and easy to update.

(PUT) **updatePermissions**(int id, String name, String description)

>Updates permissions (based on id)

>Changes the values for any non-empty parameters

>Returns result indicating success or failure

(GET) **getPermissions**(int id)

>Gets selected permissions (based on id linking to its user)

>Returns permissions information on success

>Returns indication of failure on fail

3.3 Persons

A person in the database represents a specific person in the QUEST program. This person is tied to identifying information including their name, pronoun, email address, work title, and work address. All users have to deal with Person data throughout their workflow, so accessing, manipulating, and filtering this data are necessary operations.

(POST) **createPerson**(String first_name, String last_name, String pronoun, String email, String description, String title, String work_city, String work_state, String linkedIn)

>Creates a person in the database

>Returns result indicating success or failure

(PUT) **updatePerson**(int id, String first_name, String last_name, String pronoun, String email, String description, String title, String work_city, String work_state, String linkedIn)

>Updates selected person (based on id)

>Changes the values for any non-empty parameters

>Returns result indicating success or failure

(GET) **getPerson**(int id)

>Gets selected person (based on id)

>Returns person information on success
>Returns indication of failure on fail

(DELETE) **deletePerson**(int id)
>Deletes selected person (based on id)
>Returns result indicating success or failure

(GET) **getAllPersons**()
>Returns all persons information on success
>Returns indication of failure on fail

(GET) **searchPersons**(int id, String first_name, String last_name, String pronoun, String email, String description, String title, String work_city, String work_state, String linkedIn)
>Returns all person information that matches all non-empty search criteria (parameters) on success
>Returns indication of failure on fail

3.4 Admins

Admins are the higher-ups of QUEST. Admins need to be able to publish their office location and office hours for other users to view. Admins are allowed, through their permissions, to manipulate most fields in the database such as publishing projects and managing QUEST member information.

(POST) **createAdmin**(String office, String office_hrs)
>Creates an admin in the database
>Returns result indicating success or failure

(PUT) **updateAdmin**(int id, String office, String office_hrs)
>Updates selected admin (based on id)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getAdmin**(int id)
>Gets selected admin (based on id)
>Returns admin information on success
>Returns indication of failure on fail

(DELETE) **deleteAdmin**(int id)
>Deletes selected admin (based on id)
>Returns result indicating success or failure

(GET) **getAllAdmins**()

>Returns all admins information on success

>Returns indication of failure on fail

(GET) **searchAdmins**(int id, String office, String office_hrs)

>Returns all admin information that matches all non-empty search criteria (parameters) on success

>Returns indication of failure on fail

3.5 QUESTees

A QUESTee represents a QUEST student or alumni and their identifying information that is important to the QUEST program such as major(s), graduation status, and QUEST clubs. QUEST members must be able to manipulate this data and find other QUESTees by searching through this information.

(POST) **createQUESTee**(String major, String major_2, String grad_status, String involved, String area_of_expertise, String past_internships, String quest_clubs, String cohort)

>Creates a QUESTee in the database

>Creates their cohort designation in the database

>Returns result indicating success or failure

(PUT) **updateQUESTee**(int id, String major, String major_2, String grad_status, String involved, String area_of_expertise, String past_internships, String quest_clubs)

>Updates selected QUESTee(based on id)

>Changes the values for any non-empty parameters

>Returns result indicating success or failure

(GET) **getQUESTee**(int id)

>Gets selected QUESTee (based on id)

>Returns QUESTee information on success

>Returns indication of failure on fail

(DELETE) **deleteQUESTee**(int id)

>Deletes selected QUESTee (based on id)

>Returns result indicating success or failure

(GET) **getAllQUESTees**()

>Returns all QUESTees on success

>Returns indication of failure on fail

(GET) **searchQUESTees**(int id, String major, String major_2, String grad_status, String involved, String area_of_expertise, String past_internships, String quest_clubs)

>Returns all QUESTee information that matches all non-empty search criteria (parameters) on success
>Returns indication of failure on fail

3.6 Resumes

A Resume in the database is tied to a QUESTee. Each QUESTee uploads a resume when applying to the QUEST program. As such, QUESTees (and Admins) must be able to create and update resumes as they change over time. Additionally, resumes need to be filtered to allow admins to search for specific resumes, such as resumes that have not been published yet.

(POST) **createResume**(File pdf, boolean status)
>Creates a resume in the database that is tied to the calling QUESTee
>Returns result indicating success or failure

(PUT) **updateResume**(int id, File pdf, boolean status)
>Updates selected resume(based on id linked to QUESTee)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getResume**(int id)
>Gets selected resume (based on id linked to QUESTee)
>Returns resume information on success
>Returns indication of failure on fail

(DELETE) **deleteResume**(int id)
>Deletes selected resume (based on id linked to QUESTee)
>Returns result indicating success or failure

(GET) **getAllResumes**()
>Returns all resume information on success
>Returns indication of failure on fail

3.7 Cohorts

A Cohort is a group of QUESTees based on the year they enter the QUEST program. This information allows another point for filtering by other users. Cohorts are assigned on QUESTee creation, but may need to be updated in certain circumstances.

(PUT) **updateCohort**(int id, int number, String year)
>Updates selected cohort (based on id linked to QUESTee)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getCohort**(int id)
>Gets selected cohort (based on id linked to QUESTee)
>Returns cohort information on success
>Returns indication of failure on fail

3.8 Clients

A Client represents people not affiliated with UMD that are involved with QUEST projects. Clients must maintain contact information not applicable to other user types, such as their website. Clients must be filterable to allow system users and the public to view specific client information, such as projects a client has participated in.

(POST) **createClient**(String name, String contact, String website, String description)
>Creates a client in the database with the given parameters
>Returns result indicating success or failure

(PUT) **updateClient**(int id, String name, String contact, String website, String description)
>Updates selected client (based on id)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getClient**(int id)
>Gets selected client (based on id)
>Returns user information on success
>Returns indication of failure on fail

(DELETE) **deleteClient**(int id)
>Deletes the selected client (based on id)
>Returns result indicating success or failure

(GET) **getAllClients**()
>Returns all client information on success
>Returns indication of failure on fail

(GET) **searchClients**(int id, String name, String contact, String website, String description)
>Returns all client information that matches all non-empty search criteria (parameters) on success
>Returns indication of failure on fail

3.9 Teams

A Team represents a group of QUESTees working on a project together in a course. QUESTees are assigned to a Team by an Admin or professor. Each Team is defined by its name, semester, and course. Teams must remain visible and searchable for Admins to perform administrative duties such as assigning teams.

(POST) **createTeam**(String name, String semester, String course)

>Creates a team in the database with the given parameters
>Returns result indicating success or failure

(PUT) **updateTeam**(int id, String name, String semester, String course)

>Updates selected team(based on id)
>Changes the values for any non-empty parameters
>Returns result indicating success or failure

(GET) **getTeam**(int id)

>Gets selected team(based on id)
>Returns user information on success
>Returns indication of failure on fail

(DELETE) **deleteTeam**(int id)

>Deletes the selected team(based on id)
>Returns result indicating success or failure

(GET) **getAllTeams**(int id)

>Returns all teams associated with QUESTee of the given id on success
>Returns indication of failure on fail

(GET) **searchTeams**(int id, String name, String semester, String course)

>Returns all team information that matches all non-empty search criteria (parameters) on success
>Returns indication of failure on fail

3.10 Projects

A Project in the database represents a project that a QUEST team worked on for a specific course. Projects can be created and updated by members of a team. Admins must be able to submit projects for the public to view. Once published, projects are typically filtered and searched through by all users and the general public. As such, retrieval and management of project information must be performed constantly by many users.

(POST) **createProject**(String name, String description, File poster, int status, int team_id)
>Creates a project in the database with the given parameters
>Returns result indicating success or failure

(PUT) **updateProject**(int id, String name, String description, File poster, int status, int team_id)
>Updates project (based on id)
>Changes non-empty parameters
>Returns result indicating success or failure

(GET) **getProject**(int id)
>Gets selected project (based on id)
>Returns project information on success
>Returns indication of failure on fail

(DELETE) **deleteProject**(int id)
>Deletes selected project (based on id)
>Returns result indicating success or failure

(GET) **getAllProjects**()
>Returns all projects on success
>Returns indication of failure on fail

(GET) **searchProjects**(int id, String name, String description, int team_id)
>Returns all team information that matches all non-empty search criteria (parameters) on success
>Returns indication of failure on fail

3.11 Corporations

A Corporation in the database represents a corporation associated with QUEST. This company is either affiliated with a QUESTee or pursuing information about a QUESTee. Corporations only have a limited subset of permissions and access compared to a professor unless specified otherwise by an administrator.

(POST) **createCorporation**(String name, String contact, String website, String description)

>Creates a corporation in the database

>Returns result indicating success or failure

(PUT) **updateCorporation**(int id, String name, String contact, String website, String description)

>Updates selected corporation (based on id)

>Changes the values for any non-empty parameters

>Returns result indicating success or failure

(GET) **getCorporation**(int id)

>Gets selected corporation (based on id)

>Returns corporation information on success

>Returns indication of failure on fail

(DELETE) **deleteCorporation**(int id)

>Deletes selected corporation (based on id)

>Returns result indicating success or failure

(GET) **getAllCorporations**()

>Returns all corporations' information on success

>Returns indication of failure on fail

(GET) **searchCorporations**(int id, String name, String contact, String website, String description)

>Returns all corporation information that matches all non-empty search criteria (parameters) on success

>Returns indication of failure on fail

3.12 Testing

The acceptance tests outlined in the Green Light document were carried out on the API created as a result of this project. While the exact design of the API has changed, these tests still apply. The API was also tested throughout the duration of the database and UI tests for it was used to interact between the two platforms. A full outline of the tests regarding the database and the sign offs there on can be found in the qubit/qubitapi folder in the repository.

Back To Table of Contents

4.Project Dashboard

The Project Dashboard is the user interaction and web development portion of the application. The team makes calls to the API to pull necessary information from the database as users interact with the front-end pages. The Project Dashboard ensures the front end is providing all desired functionality to users and successfully manipulating information correctly within the database.

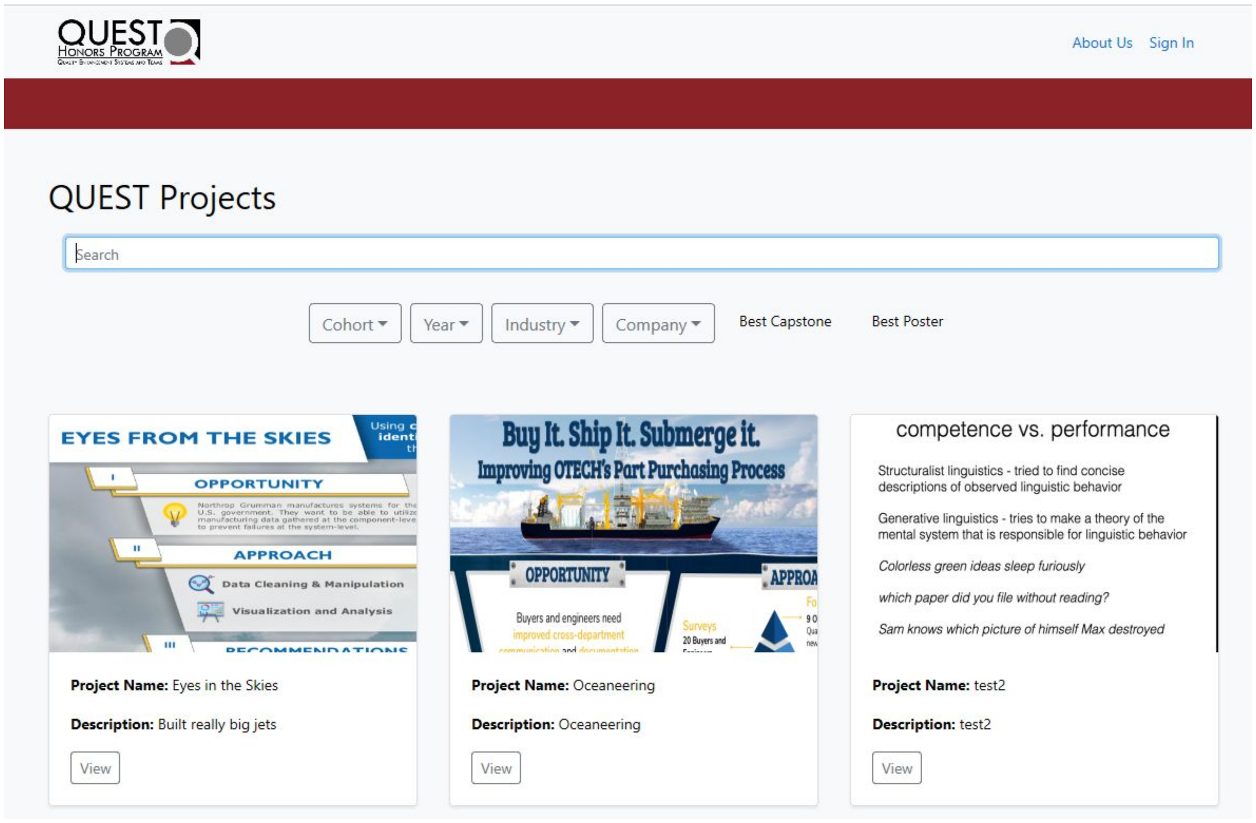
A very important piece within the Project Dashboard is that it caters specifically to the various kinds of users interacting with it. The user interface (UI) provides users with great usability and functionality regarding what the website is capable of. The project dashboard serves as the entry point for data allowing users to create accounts, input project information, search through projects, along with a variety of other functions later described. The team also developed functionality specifically for admins, allowing them to choose which projects are visible to the public and manage the users in the system. Through the Project Dashboard the team hopes to reduce the amount of manual data input done by QUEST administrators, and streamline their work process.

The Project Dashboard uses some of the technologies discussed in the API portion, mainly using Flask to communicate with databases and sort existing data on the web page itself. The CSS framework React is used to create the visual structure of the individual web pages, and SASS to compile the team's custom CSS files. The UI team is focused on creating a functional and visually appealing website that can be accessed at https://projects.questumd.com/ .

4.1 Functionality

Below is an enumerated list of all the functionality the Project Dashboard is said to have in accordance with the Green Light Document scoping out this project. Underneath each requirement, please find a detailed description of how to complete said functionality on the Project Dashboard site linked above.

Project Dashboard



Project Name: Eyes in the Skies
Description: Built really big jets

Project Name: Oceaneering
Description: Oceaneering

Project Name: test2
Description: test2

Fig 3. Project Dashboard

3.4.1.1 Clickable project cards that display project details when clicked

The project cards can be seen from the Project Dashboard main page. They consist of an image of the project's poster with the project title and description written out below. To view more information on any one project, the "View" button can be pressed taking you to the full project page.

3.4.1.2. Ability to navigate to the login page

The project login page can be reached by clicking on the "Sign In" button located in the upper right hand corner of the Project Dashboard home page as well as subpages.

3.4.1.3. Ability to navigate to the QUEST About Page

The QUEST About Page can be reached by clicking on the "About Us" button located in the upper right hand corner, slightly to the left of the "Sign In" button.

3.4.1.4. Search functionality for projects (Includes filtering)

To find a specific QUEST Project, one can either type the project title, team name, description, or other aspect of the project in the text field box at the top of the page directly below the "QUEST Projects" page title. This will remove project cards displayed below until only the ones meeting the search criteria are met. Another way to filter the projects one is looking for on the project dashboard is by using the dropdown boxes directly below the textbox. These allow the user to filter projects based on fields such as cohort, year, industry, and company. The team specifically included dropdowns for these fields since they were the most requested by the alumni and corporate partners surveyed. Additionally, projects can be filtered based on which ones won the Best Capstone and Best Poster awards by clicking on those respective buttons to the right of the dropdown boxes.

Login Page

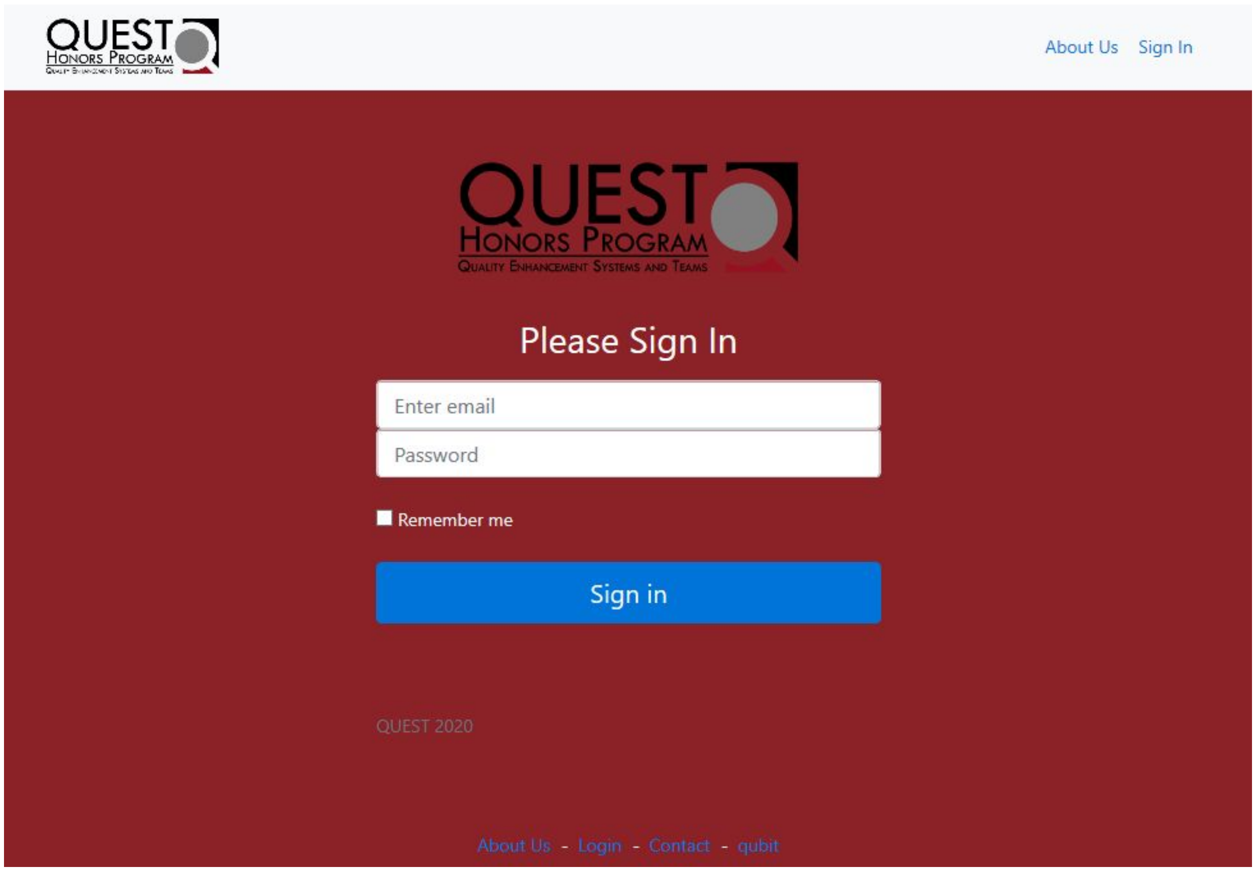


Fig 4. Login Page

3.4.2.1. Ability to login as User or Admin

Any user regardless of status can login by entering their email and passwords into the two text boxes in the center of the page. Only if the correct credentials associated with a previously created account are entered, will the signin be successful and take the user to the appropriate dashboard.

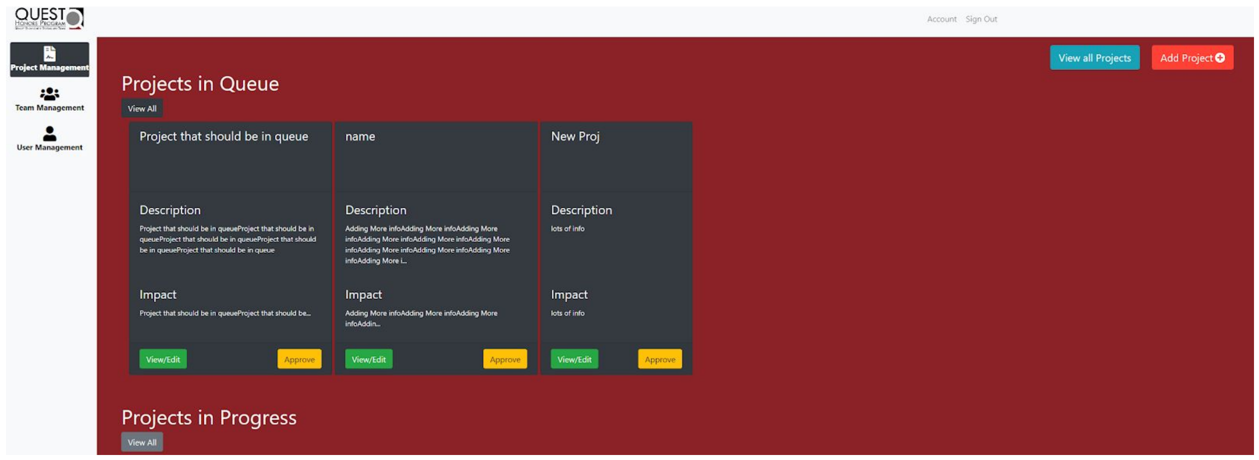
3.4.2.2. Account creation

Accounts can no longer be created via the login page, but rather must be created by the site admin who already have accounts. This is to ensure that the only people who have access to submitting projects are people who are members of the QUEST community thereby adding a level of security to the system.

3.4.2.3. Forgot password/email help

If a person with an already established account forgets their password, they must contact a site admin to be able to recover it. The functionality to recover a password via email is not implemented.

<u>Admin Dashboard Page</u>



Project that should be in queue

Description

Project that should be in queueProject that should be in queueProject that should be in queueProject that should be in queueProject that should be in queue

Impact

Project that should be in queueProject that should be...

name

Description

Adding More infoAdding More infoAdding More infoAdding More infoAdding More infoAdding More infoAdding More infoAdding More infoAdding More infoAdding More i...

Impact

Adding More infoAdding More infoAdding More infoAddin...

New Proj

Description

lots of info

Impact

lots of info

Fig 5. Admin Dashboard - Project Management Page

3.4.3.1. Publishing finished projects that have been submitted by students

To publish a finished project completed by a student, an admin should press the yellow button on the bottom right hand side of a project card. The Admin can view/edit all of the information for the respective project by clicking on the green button on the bottom left hand side of the project card prior to approving if desired. If there are more projects in the queue to be approved than room visible on the screen, additional cards can be seen by scrolling through the carousel or by clicking the “View All” button directly the “Projects in Queue” title.

3.4.3.2. Removing published projects from the Project Dashboard page

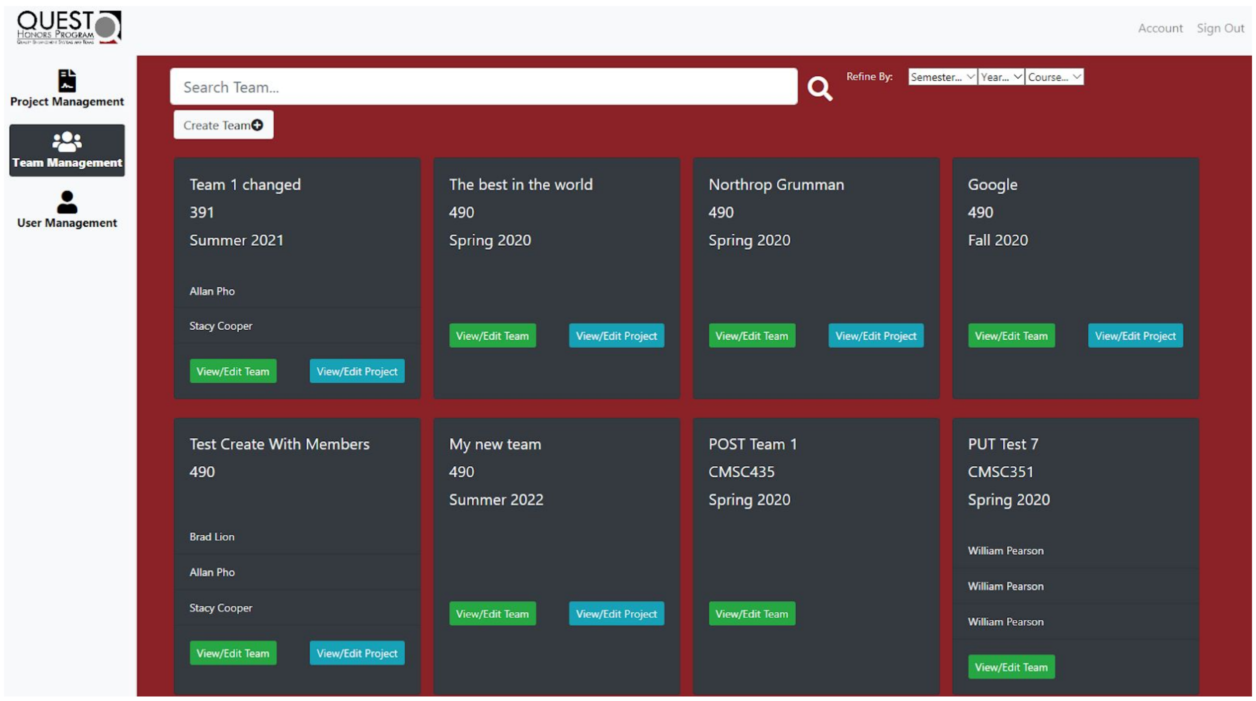
To remove a project that is already published from the Project Dashboard, the Admin needs to scroll down on the Project Management sub-page of the Admin Dashboard to view the “Published Projects” title. After this, the Admin should click on the green “View/Edit” button on the bottom left of the card of the project that they wish to remove. This will open up the project in question where the Admin can either make changes to the published project or they have the option to delete said project by clicking the red “Delete Project” button in the upper right.

3.4.3.3. Making resume books from submitted student resumes

After further discussions with Danny and Jess throughout this project, it was determined that this capability is out of scope for this project, so it is not implemented in the Project Dashboard. The functionality has been added to the API though for future implementation.

3.4.3.4. Ability to download projects with filters applied

The functionality pertaining to downloading projects has not been implemented at this time.



Team 1 changed
391
Summer 2021
Allan Pho
Stacy Cooper

The best in the world
490
Spring 2020

Northrop Grumman
490
Spring 2020

Google
490
Fall 2020

Test Create With Members
490
Brad Lion
Allan Pho
Stacy Cooper

My new team
490
Summer 2022

POST Team 1
CMSC435
Spring 2020

PUT Test 7
CMSC351
Spring 2020
William Pearson
William Pearson
William Pearson

Fig 6. Admin Dashboard - Project Management Page

3.4.3.5. Create teams

If an Admin is interested in creating teams, they need to navigate to the "Team Management" page by clicking the second button going down the left hand side of the Admin Dashboard page. The "Create Team" button can then be pressed to allow the Administrator to be able create a new team with the desired students on it. Students should already have accounts before performing this operation.

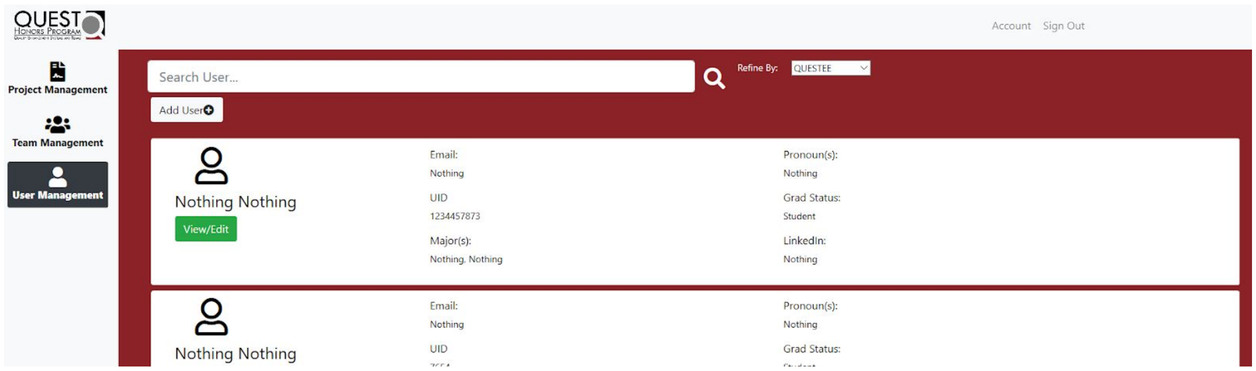
3.4.3.6. Search for projects

To be able to search for specific projects, the Admin should navigate to the "Project Management" subpage of the Admin Dashboard by clicking the first of three buttons on the left. This will show them a page similar to that shown in Figure 5. The "View All" button can then be pressed under the category of project the Admin wants to search for.

At this point, a search bar should appear at the top of the page allowing the Admin to search all projects of the same status to find the one they desire.

3.4.3.7. Vet student projects in progress

Admins can also edit student projects in process by clicking the green “View/Edit” on the bottom left of the project cards in the “Projects in Progress” subsection of the “Project Management” page.



Nothing Nothing

Email: Nothing
UID: 1234457873
Major(s): Nothing, Nothing
Pronoun(s): Nothing
Grad Status: Student
LinkedIn: Nothing

Nothing Nothing

Email: Nothing
UID
Pronoun(s): Nothing
Grad Status:

Fig 7. Admin Dashboard - User Management Page

3.4.3.8. Update users (information, teams, corporate information)

Admins can update user information specifically regarding the personal and corporate information pertaining to an individual by clicking on the third button going down the left side of the Admin Dashboard page. The Admin should then click on the green “View/Edit” button underneath the user’s name that they want to update the information form. This will take them to a page where they can then update the information for all fields. To edit the team information pertaining to a certain user, the Admin must do that via the “Team Management” subpage as displayed in figure 6 and click the green “View/Edit Team” button on the lower left of the team card.

3.4.3.9. Grant new users access to create accounts

New users do not have the ability to create accounts themselves, but Admins can create them for them. Admins are able to create users by navigating to the “User Management” subpage of their Admin Dashboard as seen in figure 7. From here they can then click the “Add User” button directly below the search text box. This will then prompt them to enter the specific information about a user and it will save said information and create an account for it as seen in figure 9. The user should then get an email with the account credentials asking them to change their password. Currently this email functionality does not work, but can be built out in the future.

3.4.3.10. Search through Users (students in each cohort, students on each project)

To search through the users in the system, the Admin should type the criteria they want to filter based on in the text box located at the top of the page. The Admin has the ability to filter based on user type (QUESTee, TA, Admin) as well through selecting an option in the dropdown menu to the right of the search textbox as seen in figure 7.

3.4.3.11. Delete Users

To delete a user in the QUEST system, the Admin needs to find the desired user they would like to delete from the User Management sub-page of the Admin Dashboard. After this, the Admin should click on the green "View/Edit" button on the bottom left of the user card. This will open up the information about the user in question where the Admin can either make changes to their information or they have the option to delete said project by clicking the red "Remove <User Type>" button in the upper right.

3.4.3.12. View relevant information at a glance on their dashboard

After discussion with Danny and Jess, the team determined "relevant information" for the administrator to mean quickly accessing information on projects, teams, and users. This influenced the page design to include a downward menu on the left hand side to include these categories to allow for easy navigation therebetween.

Student Dashboard

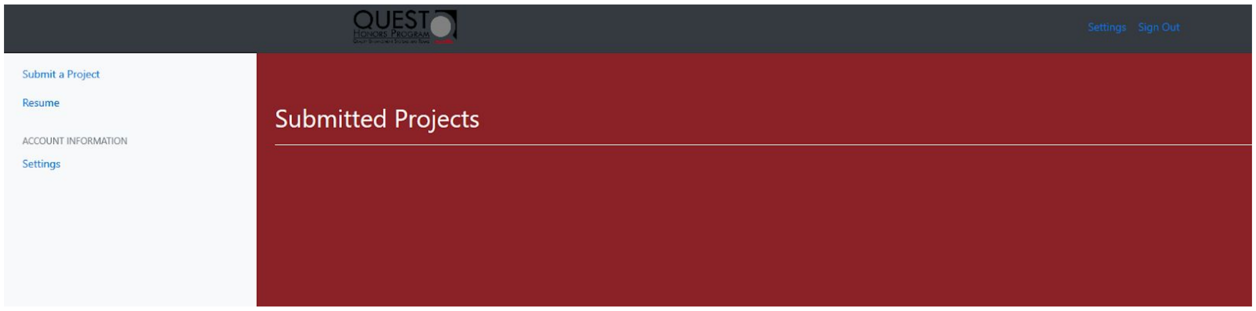


Fig 8. Student Dashboard - Submitted Projects Page

3.4.4.1. Upload, edit, and submit projects

QUESTees can submit upload projects by clicking on the submit a project button on the left hand side of their main Student Dashboard. When a project is submitted, it should appear in the body of this page as a submitted project. A user would then be able to edit this project after it has been submitted by clicking on the green "View/Edit" button located on the lower left of the project card. This functionality to view a project after the project has been submitted has not fully been implemented, so projects do not consistently appear in the current state.

3.4.4.2. Update, and edit profiles (resume upload)

While resumes were out of scope of the project based on conversations with Danny and Jess, in the future they will be able to access this functionality via the “Resume” button which is the second one down on the left side panel. Users can update and edit their profiles via the “Settings” page which can be accessed by clicking on the third button down on the left side panel as seen in figure 8.

3.4.4.3. Display user information and already submitted projects (Past and Present)

When a project is submitted, it should appear in the body of this page as a submitted project. A user would then be able to edit this project after it has been submitted by clicking on the green “View/Edit” button located on the lower left of the project card. This functionality to view a project after the project has been submitted has not fully been implemented, so projects do not consistently appear in the current state.

Student Project Submission Page

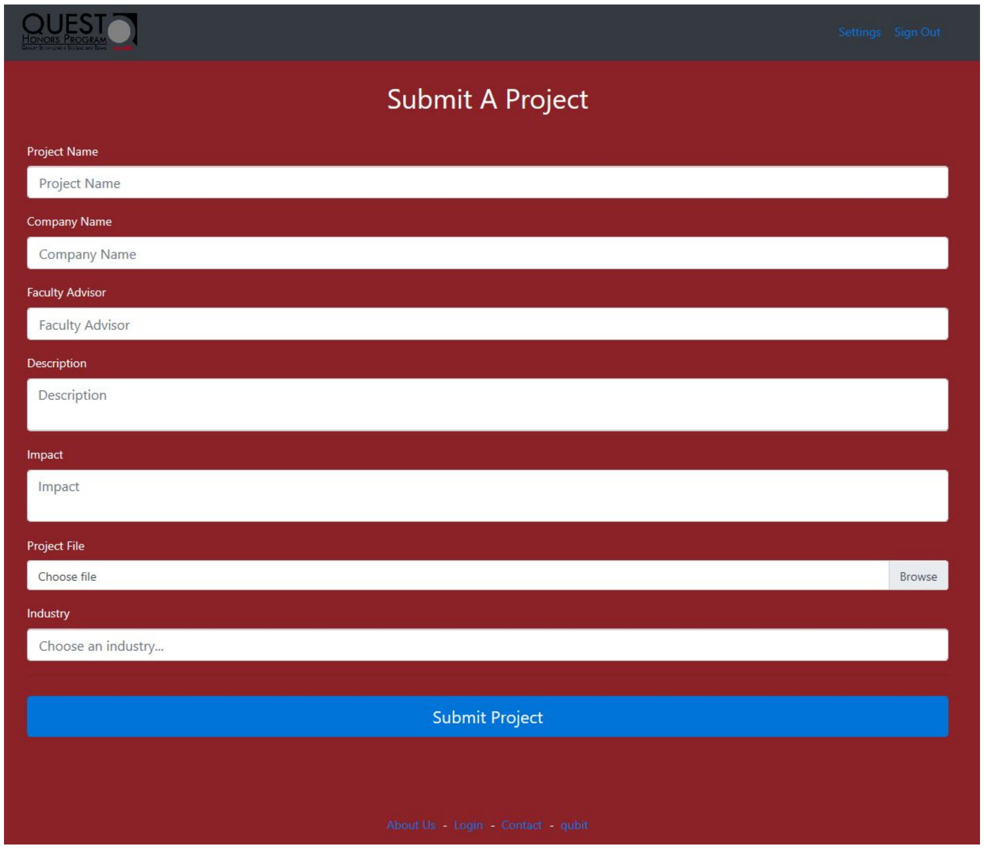


Fig 9 . Student Dashboard - Project Submission Page

3.4.5.1. Upload, edit, and submit projects

QUESTees can submit upload projects by clicking on the submit a project button on the left hand side of their main Student Dashboard as seen in figure 8. They will enter the specific project information in the form seen in figure 9.

<u>Account Creation</u>

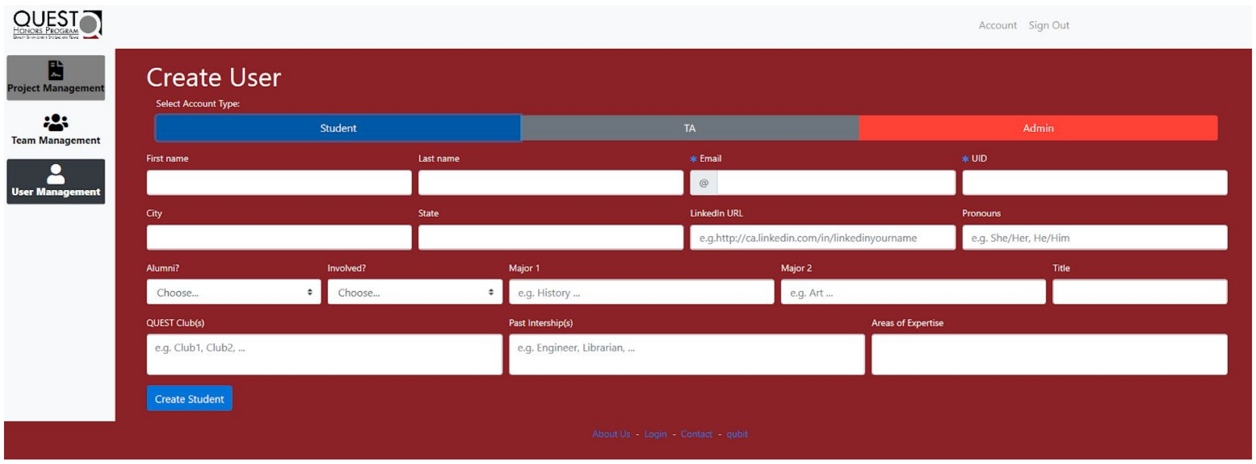


Fig 9 . Admin Dashboard - Create User Page

3.4.6.1. Ability to create user accounts

Admins are able to create users by navigating to the “User Management” subpage of their Admin Dashboard as seen in figure 7. From here they can then click the “Add User” button directly below the search text box. This will then prompt them to enter the specific information about a user and it will save said information and create an account for it as seen in figure 9. The user should then get an email with the account credentials asking them to change their password. Currently this email functionality does not work, but can be built out in the future.

3.4.6.2. Add user information and resume

User information can be added by creating an account or by editing the information of a previously created user account depicted in Figures 7 and 9.

4.2 Testing

The acceptance tests outlined in the Green Light document were carried out on the databases created as a result of this project. Visual inspections were mainly used to double check that all functionality was included. End-to-end tests were also carried out as the team added past projects to the dashboard so that the final deliverable would contain information from the past four years worth of teams as well. While the exact design of the project dashboard has charged, these tests still apply. For each of the testing procedures, API calls were used to save inputted information on the site in the database. A full outline of the tests regarding the database and the sign offs there on can be found in the qubit/projectdashboard folder in the repository.

5. Conclusion

5.1 Process Changes

To better understand the existing system that QUEST uses to track the students and projects through its curriculum, the team created a process flow diagram outlining this process. Every system that the program uses is written in a specific color. Upon looking at the diagram pictured in figure eight, it is clear that nine different platforms are used to complete the essential functionality the program bases its operations on. By understanding the current state, the team is able to identify key areas for improvement by looking into what platforms can be combined within an existing platform or whether a new platform should be introduced to consolidate the information needed.

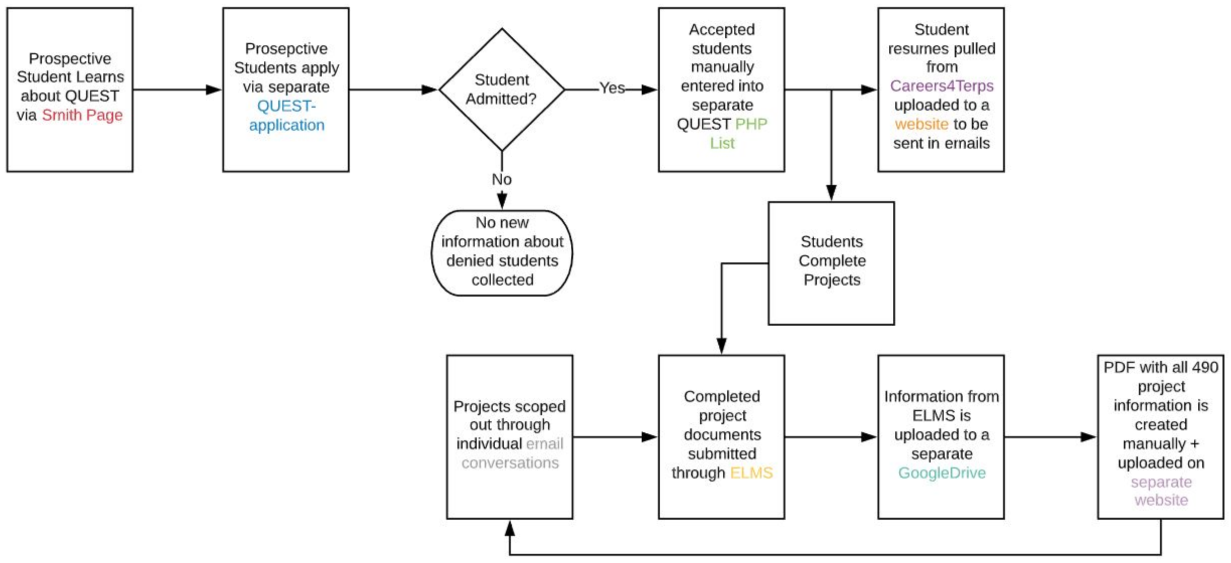

Prospective Student Learns about QUEST via Smith Page
Prosepctive Students apply via separate QUEST-application
Student Admitted?
Yes
No
No new information about denied students collected
Accepted students manually entered into separate QUEST PHP List
Student resumes pulled from Careers4Terps uploaded to a website to be sent in emails
Students Complete Projects
Projects scoped out through individual email conversations
Completed project documents submitted through ELMS
Information from ELMS is uploaded to a separate GoogleDrive
PDF with all 490 project information is created manually + uploaded on separate website

Fig 8. QUEST Existing Process Flow

After analyzing the initial state and conversing with Jessica, Danny and a handful of QUEST students, the team determined that the information about each QUEST student did not need to be pulled into a separate PHP List. While this list is currently being used to manage which email list a student ends up on, an equally functional application can be created to replace it. By pulling the information from the QUEST application directly into a new database, Jessica can forgo the manual work that the PHP List requires.

Feedback from these parties also indicated that the process of submitting final projects can also be simplified. A single Project Site that both the faculty and QUEST administrators can access would be a better platform for students to submit their final work. This singular platform would be visible to the public to better promote QUEST as a whole and to future corporate partners, while giving instructors access to be able to grade the students' work, and administrators access to be able to create the QUEST Conference Programs. While the initial state only focuses on the projects created in 490H, ideally this project site, pictured in figure nine, can be used to highlight the work done by students in all courses of the program.

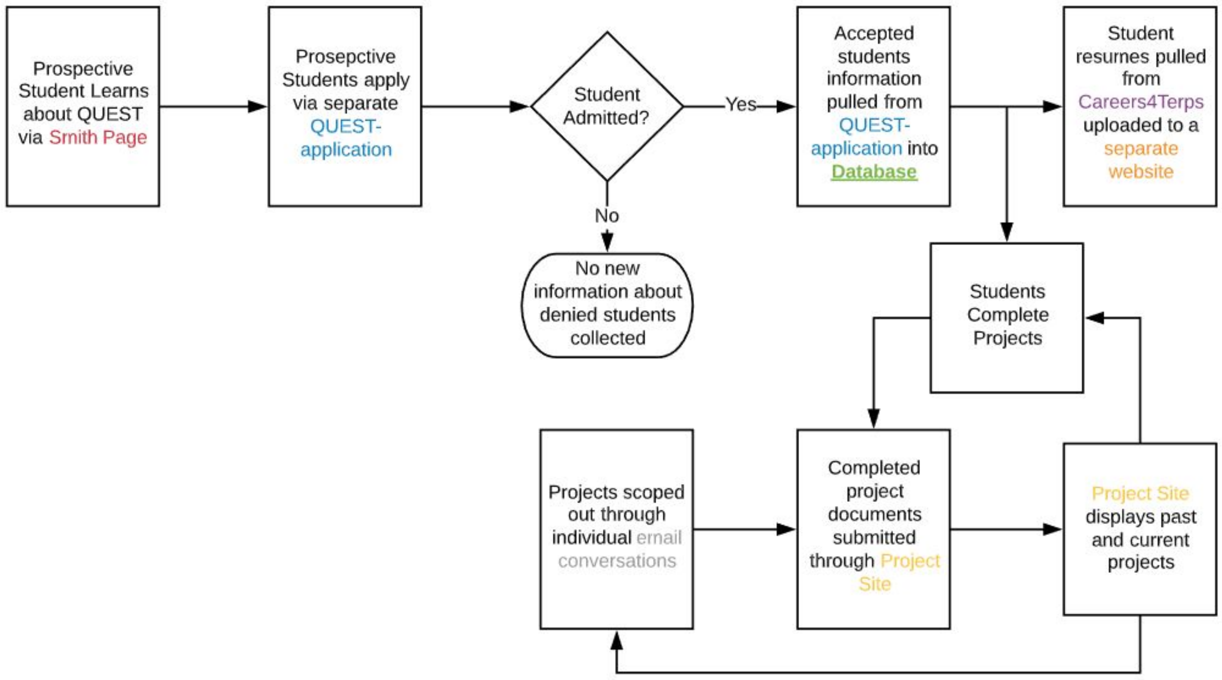

Prospective Student Learns about QUEST via Smith Page
Prosepctive Students apply via separate QUEST-application
Student Admitted?
Yes
No
No new information about denied students collected
Accepted students information pulled from QUEST-application into Database
Student resumes pulled from Careers4Terps uploaded to a separate website
Students Complete Projects
Projects scoped out through individual email conversations
Completed project documents submitted through Project Site
Project Site displays past and current projects

Fig 9. QUEST Improved Process Flow

5.2 Impact

Although the database, API, and project dashboard aspects of this project were intended to address different opportunities within the QUEST system, every aspect will make a positive impact on the program in its respective area. In terms of the database, outlining all of the information that is collected on QUEST students as they transition from prospective students into alumni in one location reduces processing time associated with entering said information in a handful of platforms. Having all the necessary information accessible through one API instead of the nine platforms required in the previous system also helps the QUEST program function in a more lean way which is more consistent with the principles it teaches its students. The quantitative impact of this single source of information can be measured as it becomes more adapted by calculating the reduction in administrative time as future cohorts advance through the QUEST curriculum.

Through the addition of the project dashboard, not only is QUEST increasing its visibility to the general public, but the program is also able to encourage other companies to become corporate partners with QUEST. The application also helps current corporate partners better understand what goes into a 490H project by providing examples of past projects. By readministering survey questions after the full adoption of the platform, QUEST can fully measure the impact by analyzing the changes in student awareness of 490H expectations prior to taking it, the utilization rates of different stakeholders, and the overall understanding different stakeholders have on QUEST. In an ideal setting, all of these numbers should increase as a result of the adoption of this platform.

This site doubles as a student tool as well for it gives current 490H students inspiration by providing many examples of past posters, and reinforcing the level of quality QUEST expects in the projects. The quantitative impact of this platform can be measured by analyzing the increase of new corporate partners who decide to collaborate with QUEST to complete projects, the decrease in quantity of emails with existing partners required to finalize a project, and how current students take advantage of the new platform. While this platform was initially built with 490H in mind, it can also be expanded to contain information for all QUEST curriculum classes to further maximize any found impact.

5.3 Future Development

While the deliverables outlined above provide a basic minimal viable product that achieves the intended outcomes, more work can still be done on the system to further encompass all of the needs of QUEST on one platform. In its current state, the teams' new system has the capabilities to add and get project information, but lacks the functionality needed to delete them. Because projects will typically only be added when they are in their final state, the delete function had the lowest priority for the team. However, as more time in the future is dedicated to this platform, this feature would be useful to have so that no one needs to go into the database directly to make project changes.

Additional areas of our website that can be built out to streamline more of the QUEST administrative processes in the future include the resume and directory functionality. While the API functionality is there and the information is stored in the database, the frontend features have not yet been added. The team ultimately hopes that the work done on this project can be used in conjunction with the work done by the Q34 Mentors' Project to create a singular intranet or one-stop-shop for all QUEST administrative needs.

6. Appendices

Appendix A. Delivery Manifest

Deliverable	How to be recognized
QUEST Unified Database	A database that passes the acceptance tests is initialized on QUEST servers at time of delivery.
QUEST Unified API	An API that passes the acceptance tests is initialized on QUEST servers at time of delivery.
Project Dashboard	A website that passes acceptance tests lives at time of delivery accessible through a link shared with the Client.
Future Development Plan	This is a folder containing documents that act as documentation for what the team has built as well as implementation strategies for other systems.
Project Advertisement	This will be visible through an about page that is accessible from the project dashboard.

Back To Table of Contents