

JMG Student Site Check-in Application

Final Report



Clients

Lanet Anthony, Samantha Brink, JMG

Developer



Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
May 6, 2022



JMG Student Site Check-In Application
Final Report

Table of Contents

I.	Introduction	4
	1. Purpose of the Document	4
	2. References	4
II.	JMG and the Desire for a Student Check-in System	5-6
	1. About JMG	5
	2. The Problem	5
	3. Overview of the Goal Product	6
III.	The Team Behind the Product	6-7
	1. Elijah Caret	6
	2. Michael Ferris	6
	3. Spencer Morse	7
	4. Xingzhou Luo	7
IV.	The Process	8-10
	1. Plan Driven/Waterfall Methodology	8
	2. Agile/Iterative Development	9
	3. V Model	9
	4. Spiral Model	10
V.	JMG Student Check-in Application Design and Planning	11-19
	1. Requirements	11-12
	2. System Design	13-17
	3. User Interface	17-19
VI.	Implementation using Budibase (and Some Minor Scripting)	19

Table of Contents (Cont.)

VII.	Testing	20
VIII.	Results	21-23
	1. Successes	22
	2. Issues	23
	3. Experience With Clients and Their Feedback	23
IX.	Deliverables	24
X.	Future Plans, Additional Information, and Concluding Remarks	25-26
	1. Security and Privacy Concerns	25
	2. Future Plans	25
	3. Concluding Remarks -Supporting JMG Students	26
	Appendix A: Document Contributions	27
	Appendix B: System Requirements Specification	28-38
	Appendix C: System Design Document	39-48
	Appendix D: User Interface Design Document	49-61
	Appendix E: Code Inspection Review	62-72
	Appendix F: Administrator Manual	73-87
	Appendix G: User Guide	88-102
	Appendix H: Project Poster	103
	Appendix I: Team Review Sign Off	104

I. Introduction (TLDR)

The JMG Student Site Check-In Application is a service that aims to automate the process of JMG students notifying teachers of their attendance at an event outside of school that is counted towards course credit. This is a capstone project for Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse in partial fulfillment of the Computer Science BS degree for the University of Maine.

1. Purpose of the Document

The purpose of this document is to describe the entire process of development of the application from conceptualization in the requirements and design phase of the first semester to the implementation and testing of the second semester.

2. References

See “*System Requirements Specification*” (Appendix B), “*System Design Document*” (Appendix C), “*User Interface Design Document*” (Appendix D), “*Code Inspection Review*” (Appendix D), “*Administrator Manual*” (Appendix F), and “*User Guide*” (Appendix G) for further information.

Salesforce. *Rest API Developer Guide*. Retrieved May 5, 2022.
https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_rest.htm

Budibase. *Budibase Docs*. Retrieved May 5, 2022. <https://docs.budibase.com/>

JMG. *What is JMG?*. Retrieved May 5, 2022. <https://www.jmg.org/about-jmg/what-jmg>

JMG. *Cast the Vision: Designing Maine’s Model*. Retrieved May 5, 2022.
<https://www.jmg.org/models-strategic-initiatives/cast-vision-designing-maines-model>

II. JMG and the Desire for a Student Check-In System

This section describes the background behind why this system is needed.

1. About JMG

JMG (Jobs for Maine Graduates) is a non-profit organization that provides support to Maine students in middle school through post-secondary education. They aim to help students with career-oriented fields such as applying for jobs, internships, colleges, as well as achieving academic success.

2. The Problem

The clients work under the ELO (Extended Learning Opportunities) program. Samantha Brink (Statewide Director) and Lanet Anthony (Manager) work with Maine students by providing them opportunities to receive credit for courses outside of the traditional school environment. These mainly encompass things such as internships, job shadowing, and community interviews. Down the line, students can use these experiences to make them stand out when applying for jobs and internships later on in life.

A lot of data needs to be stored and maintained for students involved in the program, which is why the organization uses a couple of different platforms to help them. Their learning management system, known as LearnUpon, helps keep track of student data such as grades in certain courses, attendance of events organized by JMG, or courses that they are taking under the program. They also use Salesforce as a customer relationship system. However, both LearnUpon and Salesforce don't meet the needs to track ELOs. Instead, they need an additional system that will manage these ELO experiences and allow students to use this saved data to apply for other jobs and colleges further down the road.

3. Overview of the Planned Goal Product

The goal product must deliver the required check-in functionality upfront, providing easy ways for students to check-in and for supervisors to monitor/assess student attendance. It needs to work on both desktop and mobile displays, but will likely be primarily used on mobile. Our goal product will also need to communicate with salesforce upon course creation to keep their system up to date with our activities. An important legal requirement for the app is that it follows FERPA guidelines and protects the important user information being entrusted to our app. The intended optimal student experience involves them logging in, hitting a few buttons, and being successfully checked in to whatever ELO they are attending in just a couple of minutes.

III. The Team Behind the Product

The Cyber Cookie team consists of four members, each with various distinct roles. Some of these roles were adopted from the Team Software Process (TSP) Model laid out by Laurie Williams.

1. Elijah Caret

Elijah served as the team leader, when a past fifth member of the team was going to take on the role but unfortunately had to pull out of the project. As the team leader, Elijah was responsible for maintaining the team's morale and motivating other team members to keep pursuing their tasks. Throughout the year, he has learned to practice patience and compassion for his teammates, which in turn has led to some periods of solid productivity. In addition to his role as the team lead, Elijah was also the communications liaison, in which he was responsible for communicating with the clients regarding meetings and additional issues or questions. During scrum meetings, Elijah served as the scrum master, keeping the meeting organized and attempting to follow best practices laid out by past experiences.

2. Michael Ferris

Michael serves as the teams development manager, Michael focused on making sure we were meeting the requirements and exact specifications of our client and did a lot of the heavy lifting to ensure we met those requirements. Michael ensured we were producing a high quality product by making sure good guidelines were followed.

3. Spencer Morse

Spencer served as the planning manager for the team. As planning manager, Spencer was responsible for laying out the software development requirements and assigning them to members of the team, as well as himself. Spencer helped keep the team on track to meet important deliverables as well as software development deadlines. Spencer was also in charge of hosting and maintaining the application. This meant making sure that the application was up and running to ensure asynchronous communication and development. He was also responsible for security assurance within the application. Ensuring that data was secure as well as login credentials to avoid any privacy issues and/or FERPA violations.

4. Xingzhou Luo

Xingzhou was the head of development, leading system design. He was also the version control master, handling the github repository and appropriately naming document versions.

IV. The Process

The capstone course was designed for teams to get experience with a lot of different software process methodologies. There were two different visualizations of the process as well as mainly two different methodologies implemented throughout the development of the JMG Student Site Check-In Application.

1. Plan-Driven/Waterfall Methodology

The first half of the project was much more focused on creating a solid plan and having a good understanding of what the product was going to look like. There was hardly any coding involved in this phase and a lot more meetings with the client and heavy documentation. Throughout this process, we created 4 documents that outlined an intended plan for the system.

The first document was our system requirements specification (SRS). In this document, we constructed a list of functional and non-functional requirements based on the client's proposal, and their input during the system proposal review meeting at the beginning of the semester.

Our second document was the system design document (SDD). Details of the system architecture and its components were established here. Overall, it went through a major change throughout the process, which will be discussed later on in this document.

The third document was the user interface design document (UIDD). Here, the focus was solely on designing how the user would interact with the system. Laying out the details in this document was incredibly important since one of our requirements, that our user interface was simple and easy to use, was of highest priority.

At the end of our planning phase, we compiled what we had into another document, called the critical design review document. The goal of this document was to show our client our entire plan so that they could either approve or bring any changes that they want. This was paired with a presentation labeled the critical design review.

The first three documents were also reviewed with the clients after their creation, the clients would show their approval by signing off via an appendix towards the end of each document.

2. Agile/Iterative Development

The second half of our project was much more focused on development and coding. Here, we followed a more agile approach. What this meant was that we iteratively developed our product. We broke it down into sprints that would last two weeks, starting each Wednesday. Each sprint had a goal. For example, our first sprint consisted of launching a Budibase hosting portal, and beginning the scaffolding for the database. Each sprint was kicked off with a scrum meeting. The scrum meetings served as a way to establish the results of the previous sprint, as well as plan the next sprint. We discussed tasks that were finished, tasks that weren't finished, and what new tasks needed to start being tackled. We were originally going to create some backlogs, but the system was simple enough that documentation of each meeting, as well as biweekly progress reports from each team member were enough to track our progress.

3. V Model

The V model was mainly used as a way to represent the overall scope of what we were focusing on relative to the product. Each end of the V represents the highest level or the entire product where this final report was, acceptance testing was conducted with the clients present, and requirements were laid out for the system at the beginning of the semester. As we work our way towards the center/bottom of the V, we begin breaking the system down into more components with system design and integration testing. Eventually, we reach the center of the V which represents the actual programming portion of development. Essentially, the layers of the system are stripped down and then reapplied. Its was also used to help us understand where we needed to be in the process and what documentation needed to be completed at that time.

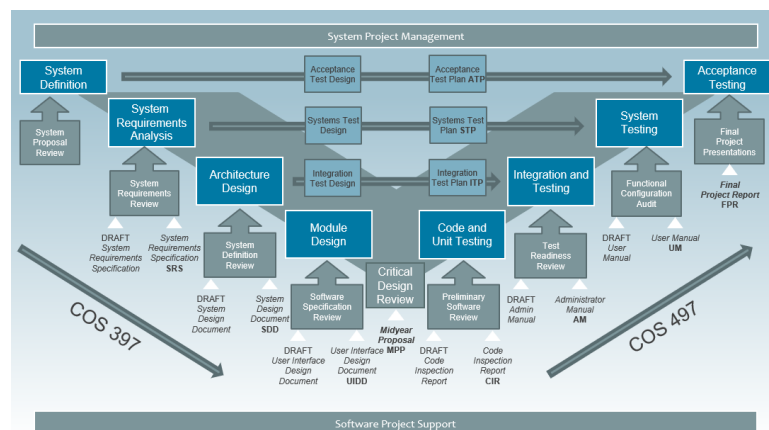


Figure 4-1: V Model - This helped lay out where we were at and where we were headed next.

4. Spiral Model

Another model that we used was the spiral model, this was used more in tandem with our agile methodology during the development phase of the project. The biggest part of it was that each circular revolution in the spiral represented an individual sprint where the goals laid out at the beginning of that revolution were due to be met. Like the V model, it gave us a reference as to where we needed to be in terms of progress.

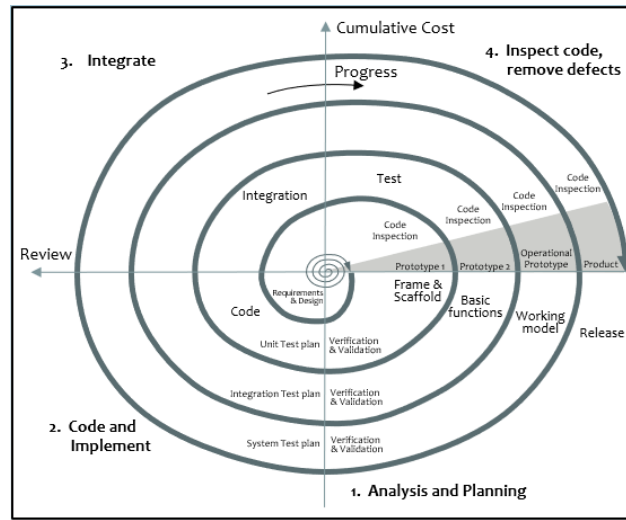


Figure 4-2: Spiral Model - This was used to help us track where we were and where we should be regarding the development phase of our project

V. JMG Student Check-In Application Design and Planning

This section describes the overall process of planning the application's design and functionality.

1. Requirements

After reviewing the proposal and discussing further details with the clients, the system's requirements were developed to help define the application's functionality and performance standards. You can see the full document under appendix B of this report.

a. Functional

The following functional requirements are descriptions of services that the system will offer that were presented and requested by the client in their proposal.

- The application must protect user data as defined by FERPA guidelines.
- The application must provide a check-in button on the homepage for students.
- The application must provide a feedback form for Supervisors to access .
- The application must provide a means for roles to search for users and courses under their jurisdiction.
- Supervisors must be able to verify student check-ins
- Advisors must be able to view their respective student's check-in reports, including feedback provided by supervisors.
- The system should send its data to JMGs Salesforce server
- All users and courses should have a unique ID through which they can be identified.

Some of the requirements being highlighted here are that the system will comply with FERPA guidelines, which is essentially a Federal Education Act that allows parents to access their child's student records for children under 18. The system will also have a simple check in button as it is the main functionality of this system. Another feature the client wanted was a feedback ability where the site supervisor can assess and evaluate the student's performance. The system will allow for this and include a button the student may click to review their performance. The system will also include roles based on if you're a student, site supervisor, JMG advisor, or admin which will control what data they are able to view, access and manipulate.

The client is in collaboration with a company called Salesforce and therefore the system will allow data and course creations to be communicated with Salesforce. Lastly, each student will be given a unique identification number aside from their email address and password to better help protect and also organize student data. These are a few of the functional requirements that we believe to be of the highest priority for this system.

b. Non-functional

These requirements help define the systems performance metrics; the ones listed here were ranked highest priority.

- The application must not take more than 3 page links from the home screen to get to any content.
- The application must fit both mobile and desktop displays.
- The site will allow admin accounts the ability to delete any user or course on the site.

One of the largest concerns for the client is functionality and ease of use. This app is designed to be used by younger students who are on the go. This means that the application will have to be able to fit and run nicely on any size display, especially mobile. The application will also not take more than 3 page links from home to reach any content as ease of use is a high priority in this system.

c. Testing plans

A test was planned to be conducted on the application using a small sample of students in the JMG program sometime in the second half of the Spring 2022 semester. We then planned to create something similar to a focus group, where we would receive some feedback from the sample of students, supervisors, and advisors after the trial period had finished. This input then was to be considered and changes would be made based on that feedback.

Other testing of our code and system will be described more in detail in the code review section of this document.

2. System Design

The application will be constructed as a web application, and therefore must have some defining features such as a front-end, back-end, and database. This section will describe the major components, how they interact with each other, and additional supplemental software that we plan to use. An important note is that our overall design had a major change after the system design document was finalized, this will be explained in more detail throughout this section. The full document itself can be found on appendix C of this document.

a. High-Level Architecture

At the highest level, the original design of the system consisted of 4 major parts interacting with each other.

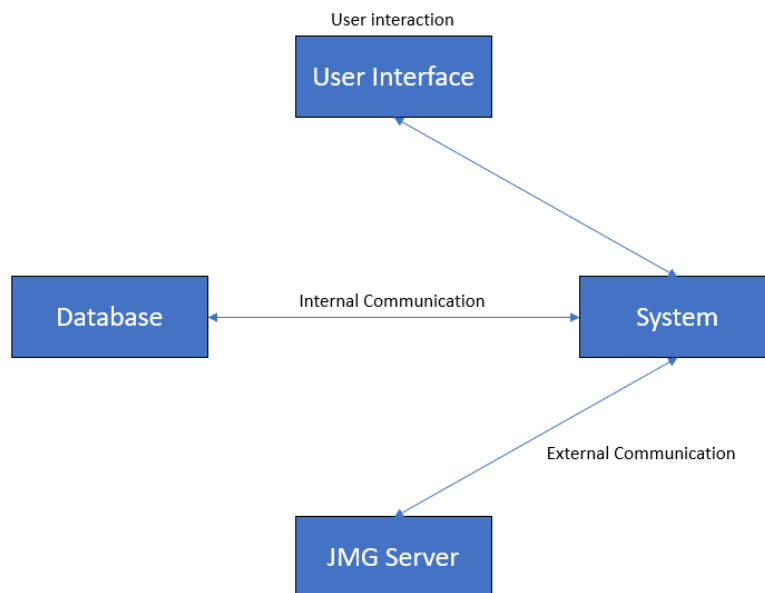


Figure 3-2-a - System Architecture Overview - Four major components are shown above, including the user interface, the system, the database, and the JMG server.

The user interface constitutes the application's front end and serves as the interaction point between users and the system. On the back end, the main system and the database will communicate internally to perform most of the applications functionality including managing user accounts and student check-ins. We also planned to have the system communicate with the JMG Server (Salesforce) in order to send data. However, as we got further along the project, we realized that it was no longer necessary.

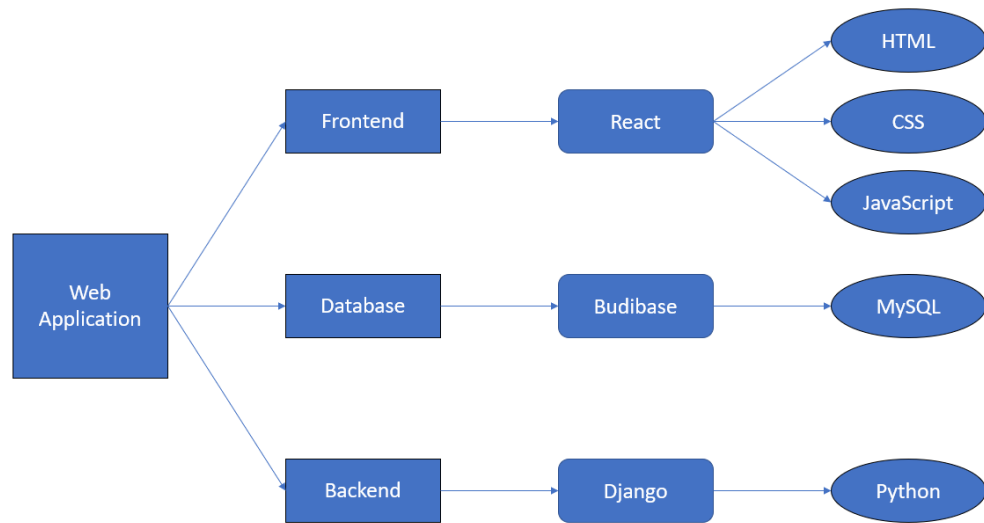


Figure 3-2-b - Technical Architecture Overview – The web application is split into three logical components, with the rectangle shape representing components, the round rectangle shape representing platforms, and the ellipse shape representing programming languages.

Looking at the former design of the system in more detail, figure 3-2-b shows the components of the front end, database, and back end that we plan to use. The frontend component features a user interface that the user can either interact with on desktop or mobile platforms. Our choice for the frontend platform was React. React is among the most popular of the JavaScript libraries for building UX design, featuring declarative development view and encapsulated object components.

The database component contains and organizes the user data. The original plan was to use a mySQL server, then a postgresQL server, but eventually we found that an external database was not necessary further along development. Our choice for the database platform was Budibase, which is further described in section 3-2-c, as that was the main platform used in our new design. The backend component hosts the core functionality of the system. It is responsible for the communication between the user and the main server. Our original choice for the backend platform was Django. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

b. Database

There is a small, but important collection of data that needs to be kept for each user.

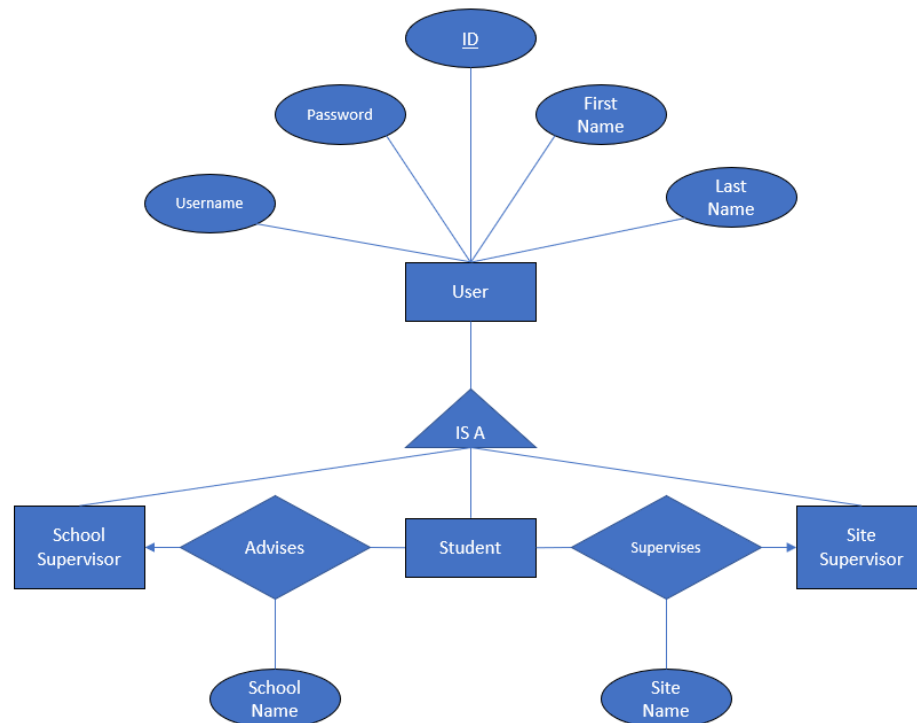


Figure 3-2-c - User ER Model - This model is used to help describe user data and make a distinction between various kinds of users

Each user will have some basic information to help identify them. Their first name, last name, identification number, username, and password will all be stored in this database. The user's specified email may also need to be stored especially when they are trying to register a new account. Their identification number will serve as the relation's - essentially the "table" - key, since every ID number is unique. Figure 3-2-a also points out the distinctions between a student, school supervisor, and site supervisor since they all have different functions within the application.

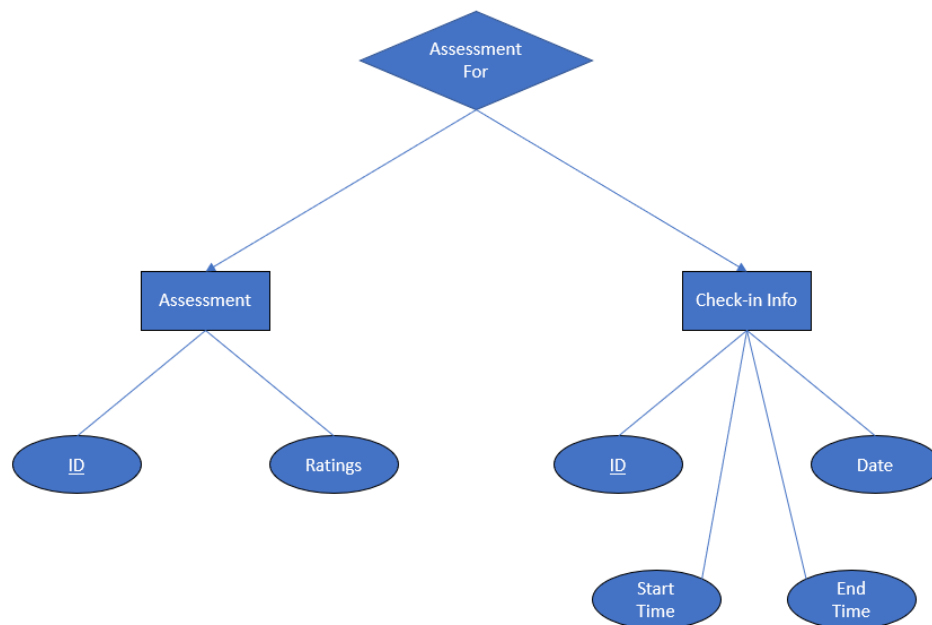


Figure 3-2-d - Check-In Data ER Model - This model contains the data stored for each student check-in

The data collected for each check-in includes a performance assessment - filled out by the site supervisor - and some additional check-in info. Each assessment will have an identification number as well as the various ratings that the site supervisor will judge the student's performance on. Each check-in will have an identification number, the date of the check-in, and the start and ending time, which will be needed in order to know whether a check-in is valid. We eventually paired the check-in info with the assessment.

c. Salesforce and Budibase

As mentioned in section 2-a, Salesforce and Budibase are platforms that we intend to interface with (Salesforce) and use (Budibase) in our design. Salesforce data can be accessed through the REST API. The API serves as a very useful package to help access RESTful service - to which Salesforce falls under - data.

Budibase is a low-code platform created with the intention to simplify the process of coding an application's backend. It has a plethora of useful features including its own "pre-packaged" database. It is also compatible with various relational database styles such as Postgres and MySQL. There are a variety of options we can choose from when it comes to implementing the database. They also provide support for implementing REST data sources which is necessary for us to link the application to JMG's Salesforce server. Budibase has a very helpful docs page that provides basic tutorials on building a CRUD application. CRUD is an acronym that stands for the four main actions of a web application: create, read, update, and destroy, which are the main actions that we are most concerned with when it comes to programming the application's functionality.

3. User Interface

The user interface was designed with simplicity and functionality in mind, with the intent to bring the user as close as possible to the utility they need. This means that the page structure and available navigation will change based on the user role. In general, almost any page on the site can be reached directly from the home screen, with the single exception of the attendance page for teachers and supervisors requiring the student lookup page to be used. The navbar is shown in the bottom left of figure 3-2-e and will appear on every page following log in.

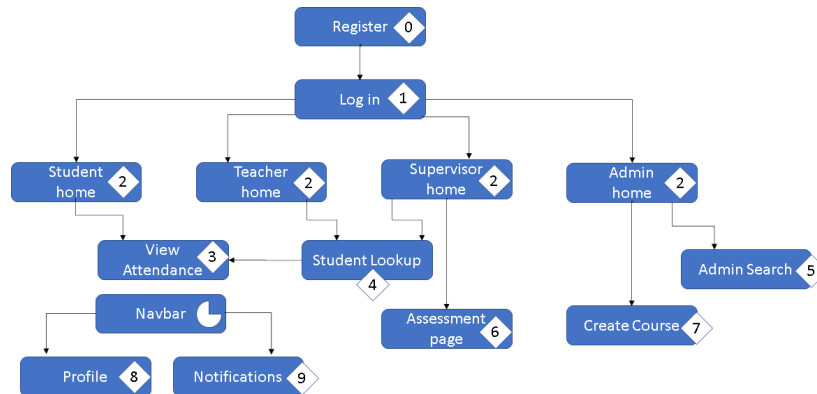


Figure 3-2-e. Navigation flowchart

The Login and Registration pages are the necessary prerequisites for accessing the site and include basic information required for the creation or validation of an account.

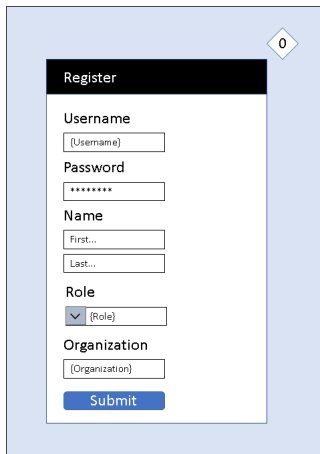
A registration form titled "Register" with a diamond icon containing the number 0. It includes input fields for Username, Password, Name (First and Last), Role (a dropdown menu), and Organization. A blue "Submit" button is at the bottom.

Figure 3-2-f - Registration

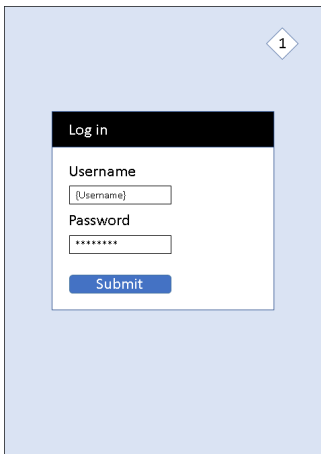
A log in form titled "Log in" with a diamond icon containing the number 1. It includes input fields for Username and Password, and a blue "Submit" button.

Figure 3-2-g - Log In

The home screen is the main point of divergence between different user roles, the student (figure 3-2-g.) has a button for checking into a particular course and another to view their attendance (figure 3-2-h.) in the previously mentioned course. Meanwhile supervisors and teachers are both given access to a student lookup feature (figure 3-2-i.) for the courses they have jurisdiction over, from which they both view a selected students attendance page (figure 3-2-i.), with the added feature for the supervisor to write an assessment of the given student.

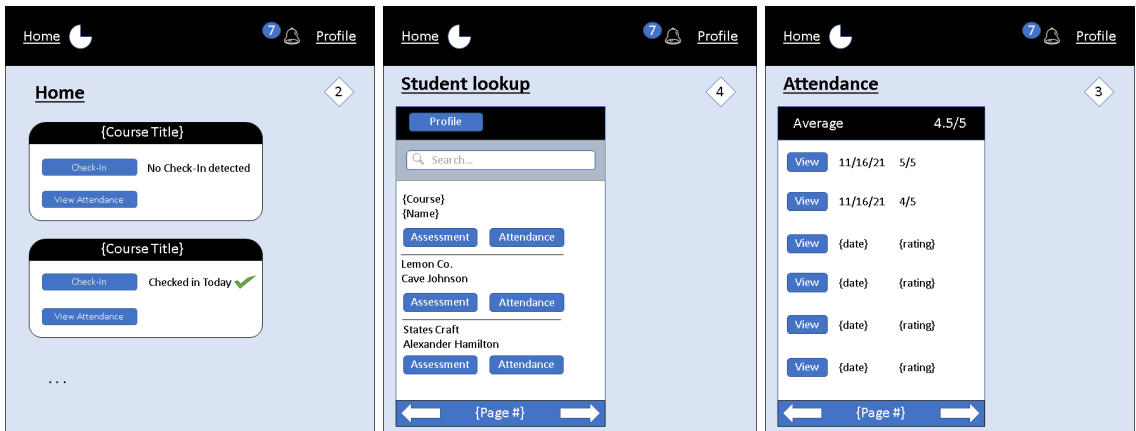
Three screenshots of the application interface. The first, titled "Home" with a diamond icon containing the number 2, shows two course cards with "Check-in" and "View Attendance" buttons. The second, titled "Student lookup" with a diamond icon containing the number 4, shows a search bar and a list of students with "Assessment" and "Attendance" buttons. The third, titled "Attendance" with a diamond icon containing the number 3, shows a table of attendance records with "View" buttons and a page number at the bottom.

Figure 3-2-g – Home Figure 3-2-h - Student Lookup Figure 3-2-i - Attendance

The final important divergence in site functionality is in the admin account, which gets access to both course creation privileges (figure 3-2-j) and an admin search (figure 3-2-k.) capable of browsing both users and courses to delete them or view the pages from the perspective of any other user.

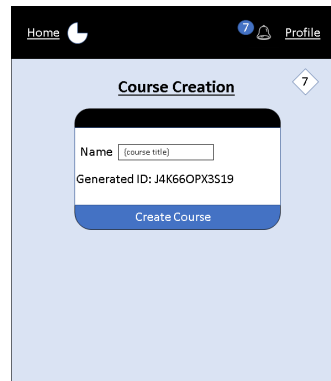


Figure 3-2-j - Course Creation

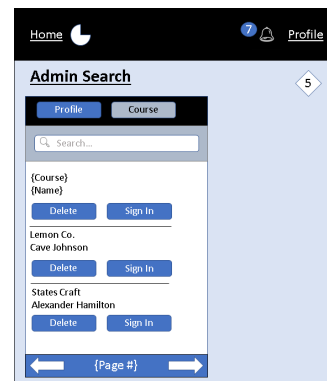


Figure 3-2-k - Admin Search

VI. Implementation using Budibase (and Some Minor Scripting)

The old design was dramatically changed, reducing the number of open source platforms and frameworks to just one: Budibase.

Budibase is a low code platform designed to aid in the production of web applications. It gave the team everything that was needed for this type of application. It had a built-in database, UI design module, as well as an automation module. The need for docker containers and web servers was no longer needed.

A major aspect of using Budibase, in addition to its all in one presentation, was its ease of use for administrators, who may not be as technically sound as a software developer. A lot of the required maintenance boiled down to deleting users, adding users, assigning roles, and backing up data, all of which can be done very easily via Budibase. More detail on system maintenance can be found in appendix F.

Budibase's UI model allows for custom CSS and Javascript which help make formatting was displays more customizable. The UI also allows information to be displayed via bindings, which are done using language known as handlebars.

VII. Testing

Testing of a system is a crucial part to any successful application. Since we used a low code platform to create our application, we didn't require much actual code review. Instead we tested our system as follows.

We started out by creating a defect checklist. This list would include about 16 common defects or problems that could be present in our application. We rated these defects on a scale of critical, major, or minor. This helped us know what kind of potential errors/defects we should be looking for during our inspection. These defects ranged from issues like logic errors and UI defects, all the way to issues like permission/access control and security oversights.

As mentioned above, there was minimal coding that went into this application so to start out testing, we broke the system down into modules and tested each one individually. These modules include:

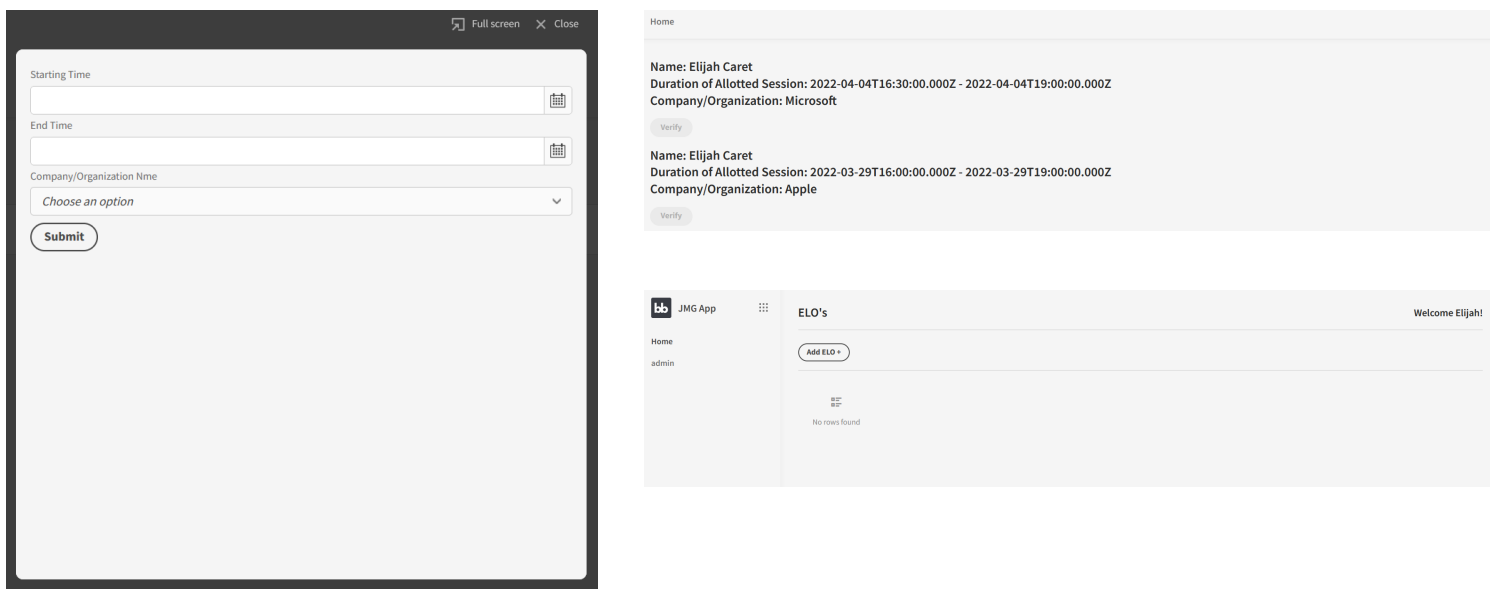
- **Check-in:** this module allows the student to perform check-in action.
- **Check-in Verification:** this module allows the instructor to confirm a student's check-in action.
- **Portal UI:** this module provides the user with a graphical interface that interacts with other pages on the site.
- **Course/ELO Adding:** this module allows the student to add courses/ELOs to their schedule.
- **Course Enrollment:** this module allows the instructor to add students to a course.
- **User Database:** this module stores the data of the site.
- **Check-ins Database:** this module stores the check-in records.

We then tested each module thoroughly and reported any defects or problems we found into a tabular format. This table, found in *Appendix E*, listed the description of the defect, the module it was found in, and the defect category it belongs to. We then used that information to fix and resolve those defects that were found during the testing meetings.

Further information on testing can be found at *Appendix E: Code Inspection Review*.

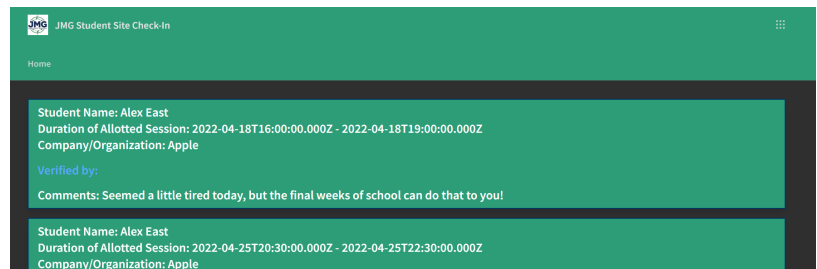
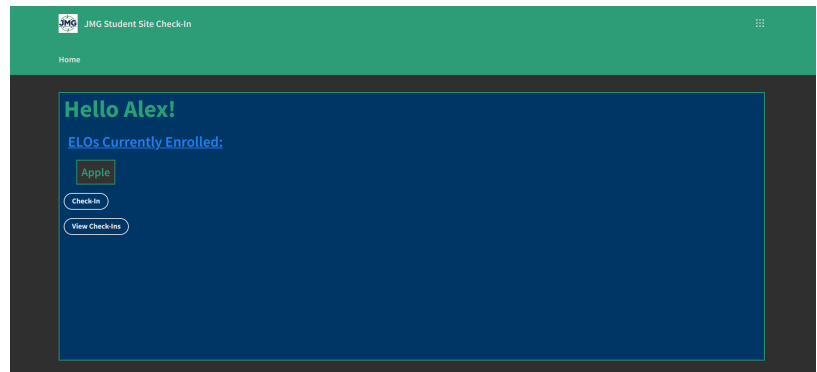
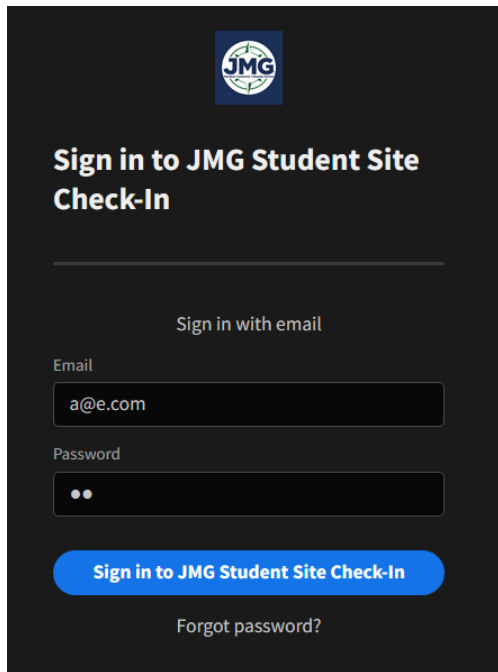
VIII. Results

Version 1.0: On April 14, 2022 the application was tagged and released as Version 1.0. With this version we were able to release most of our major system requirements. Students can sign in with their email and given password. From there they are prompted to change their password to one of their choosing. They then have access to check in to any ELO's they are enrolled in. From there, the application allows advisors to verify student check ins, as well as leave feedback on students. These components were the largest and most important aspects of the application and can all be found in version 1.0. More information on what can be accessed through version 1.0 of the application can be found in *Appendix G: User Guide*.



Figures 7-1, 7-2 and 7-3: Screenshots of V1.0 - This was the initial beta release of the product

Version 1.01: Version 1.01 was tagged and released on April 25th, 2022. The team managed to make some more additions to the product, although still not ready for a full official release, additional functionality was added including advisor views and feedback. The UI was given a small upgrade as well with a different color scheme as well as minor changes to locations of various containers.



Figures 7-4, 7-5, and 7-6: Version 1.01 - This version includes some slight changes

1. Successes

We believe that it was the right call to stick with just Budibase to develop our application, as the learning curve to figure out Django, and then setting up a web server was too steep for the amount of time that we had, perhaps if we were able to dedicate our whole days to this project, we would have went with the more customizable aspects of our original system design.

We were able to implement a basic form of version control and CI/CD, where any changes made could be saved and put into a new build, as well as going back to a previous build when needed. Everyone was also able to test their own components before integrating them together easily enough, as everyone on the team created their own stub application to work on their assigned tasks without breaking the major build.

When the team was working together on the project, a lot of fruitful productivity was there and a lot of progress was made on whatever task we were focusing on such as documentation and development.

Our app has the essential functionalities that the clients labeled as highest priority. This is something that they can take and push further along or, at the very least, have a better idea of what they want.

2. Issues

Testing could have been more expansive and thorough. Although code inspection really helped us a lot, our integration phase was not very well tested. The application was simple enough however that any issues could be fixed quickly and nothing really had the potential to break the entire system.

There were some very unproductive scrum meetings due to various team members not being present. Perhaps if we were actively engaged and present for every meeting, more progress would have been made by the advent of the version 1.0 release.

It may have helped give us a more tight guide to the goal product if we met with our clients more, in addition to the required meetings for documentation reviews. The required meetings also had some team members not present which also led to a break in understanding when it came to the document due dates.

3. Experience with Clients and Their Feedback

Samantha and Lanet were very communicative and understanding of our whole process, every meeting with them was productive and led to either them having a deeper understanding of the product or the team having a deeper understanding of what they wanted. They have expressed how wonderful it was to work with the team and that they really took note of how active the team was when it came to listening to what they wanted.

IX. Deliverables

All of the current up to date versions of our documentation and deliverables can be found on our GitHub Repository: <https://github.com/clatosklisto/CyberCookie> .

Our deliverables include:

- System Requirements Specification (Completed October 25th, 2021)
- System Design Document (Completed November 10th, 2021)
- User Interface Design Document (Completed November 29th, 2021)
- Critical Design Review Document (Completed December 13th, 2021)
- Code Inspection Review (Completed March 9th, 2022)
- Administrator Manual (Completed April 3rd, 2022)
- User Guide (Completed April 20th, 2022)
- JMG Student Site Check-in Application V1.1 Textfile (Completed April 25th, 2022)

Currently, the application is hosted locally with a reverse proxy service, making it available on the web through a masked link. However, the application is ready for more permanent hosting as it can simply be downloaded into a text file. This file can then be sent to the computer/server that JMG will be hosting from, and once they set up their BudiBase portal they can simply import the text file and the application will be made.

Maintenance of this system is further explained in *Appendix F: Administrator Manual*.

X. Future Plans, Additional Information, and Concluding Remarks

1. Security and Privacy Concerns

Since the application is designed to be a tool that maintains educational records for students under the age of 18. Therefore, as described in section 3-1-a, it must be in compliance with FERPA (Family Educational Rights Privacy Act). The application must be designed so that parents are able to access and see their child's performance data. However, we must make sure that students can not see other student's data. Budibase mostly covers these concerns since it is built with security in mind. We also plan to dump any data in the system every 5 years, as it should be maintained within JMG's Salesforce dataset.

The data kept on each student is very limited however, where the only personal information that is maintained is very basic such as their name as well as the school that they attend.

2. Future Plans

Our future plans consist of polishing up our work, finishing the final details, and then assisting with maintenance and deployment. Going forward we will assist JMG in importing and hosting our work through Budibase and the cloud platform digital ocean. We will be there to answer any questions they might have and to assist them in whatever way possible during the roll out of the new JMG check-in application.

3. Concluding Remarks - Supporting JMG Students

The JMG Student Site Check-In application will aim to help students achieve success in their future careers. This will be done through giving them data on their performance at their ELOs, to help reflect and improve on it, as well as giving them a reference/piece of evidence to which they can use on various applications and resumes to improve their chances of success in applying for other jobs, internships, and colleges. We also want the application to be an opportunity for these students to receive the best support possible through functionality provided for site supervisors as well as school supervisors, because success doesn't typically happen without a lot of support and help.

We hope that our application will help students in the JMG program to thrive and that through our cooperation our clients learned more about what it really is that they wanted in a product.

Appendix A: Document Contributions

- Elijah Caret:
Part I, Part II, Part III-I, Part IV, Part V, Part VI
- Michael Ferris:
Part X
- Xingzhou Luo:
- Spencer Morse:
Part VII, VIII, IX, Appendix E-G

Appendix B: System Requirements Specification

JMG Student Site Check-in Application

System Requirements Specification



Client

Lanet Anthony, Samantha Brink, JMG

Developer



Elijah Caret, Michael Ferris, Xingzhou Luo,
Spencer Morse

University of Maine
October 25, 2021
Version 1.0

JMG Student Site Check-In Application
System Requirements Specification

Table of Contents

I.	Introduction	30-31
	1. Purpose of the Document	30
	2. Purpose of the Product	30
	3. Product Scope	30-31
	4. References	31
II.	Functional Requirements	31-37
III.	Non-Functional Requirements	37
IV.	User Interface	37
V.	Deliverables	38
VI.	Open Issues	38

I. Introduction

The JMG Student Site Check-In Application is a service that aims to automate the process of JMG students to notify teachers of their attendance at an event outside of school that is counted towards course credit. This is a capstone project for Elijah Caret, Michael Ferris, Xingzhou Luo, Spencer Morse, and Brennan Schatzabel in partial fulfillment of the Computer Science B.S. degree for the University of Maine.

1. Purpose of the Document

The purpose of this document is to lay out the requirements for a student check-in application for the clients (Lanet Anthony and Samantha Brink from JMG) created by the development team (Cyber Cookie). It serves as a complete overview of exactly what the application will do upon completion.

This document contains functional and non-functional requirements, the tests that will be performed to ensure that the requirements are met, a list of deliverables, and any remaining open issues

2. Purpose of the Product

One of the responsibilities of the JMG program is to track and report the attendance of students at out-of-school positions, such as internships. JMG must be able to report this information back to the students' school and the out-of-school position needs to be able to confirm the attendance reported by the student. The JMG's LMS (learning management system) does not provide this functionality however, creating a problem for the JMG.

In order for the JMG to meet its responsibilities they will need an application that can keep track of self-reported student attendance, allow for an out-of-school overseer to rate their participation, and be visible to both teachers and JMG admins. This application will need to be web based and accessible via desktop and mobile devices. Additionally, the delivered product must interface with the salesforce API and preferably also the Learn Upon LMS API.

3. Product Scope

The system will be a web application. Included is the backend software (Python or other language) to help with making database queries, interpreting user actions, as well as sending and accessing data from Salesforce. Another piece is the front end, which will encode the user interface and look of the web application, this also how the user will

interact with the system. Finally, there will be a database (SQL or some relational database package) that will hold data such as login information.

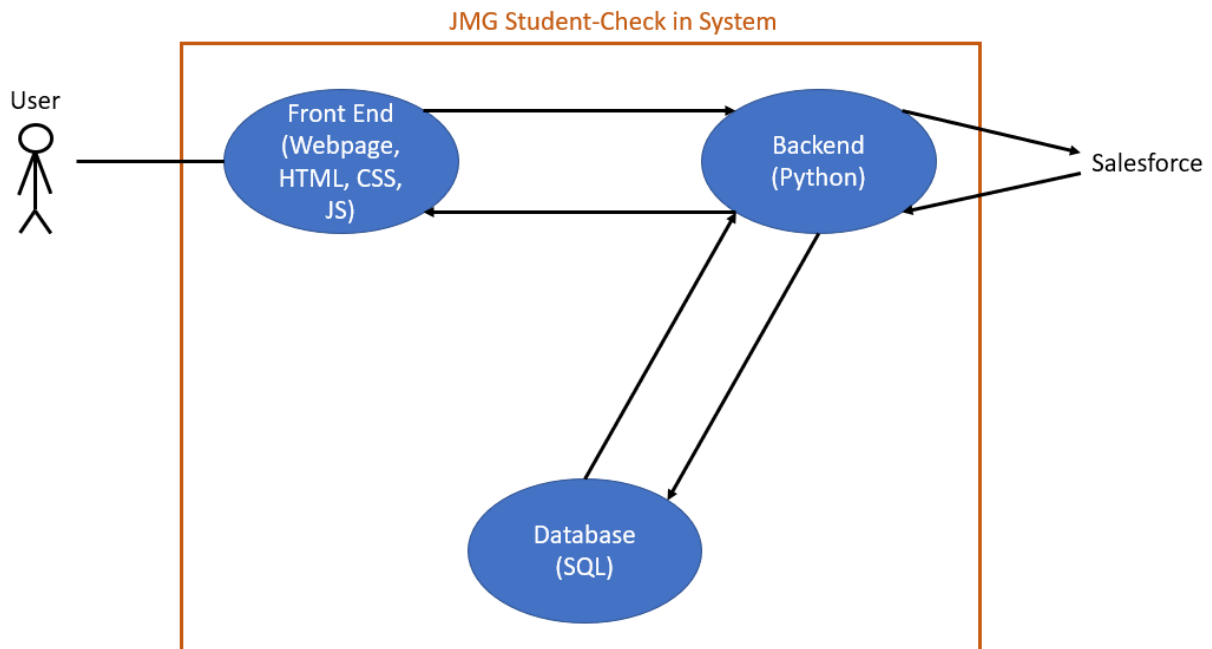


Figure B.I.3: product scope overview

4. Reference

Salesforce. (n.d.). *REST API Developer Guide*. Developer Portal. Retrieved October 26, 2021, from https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_rest.htm.

Martin, M. (2021, October 6). *Functional requirements vs non-functional requirements: Differences*. Guru99. Retrieved October 26, 2021, from <https://www.guru99.com/functional-vs-non-functional-requirements.html>.

US Department of Education (ED). (2021, August 25). *Family educational rights and privacy act (FERPA)*. Home. Retrieved October 25, 2021, from <https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html>

II. Functional Requirements

Based on the proposals and proposal review meeting with the client. These functional requirements were developed to encapsulate the basic functionality of the system.

1. The user shall be able to create a new account with a credential combination.
2. The user shall log in to the system with a credential combination.
3. The system shall verify the user's credential combination when logging in.
4. The system shall exchange data with Salesforce via the REST API.
5. No user shall be able to view another user's data except for the system administrator.
6. The system shall allow the student user to check in with a button click.
7. The system shall notify the student user whether the check-in was successful.
8. The system shall notify the site supervisor when a student user has checked in.
9. The system shall allow the site supervisors to fill out a form to provide feedback on student performance.
10. The system shall allow the school supervisor to view the site supervisor's feedback for its students.
11. The system shall allow the school supervisor to check student users in when they forget to check in themselves.
12. The system shall notify the school supervisor whether a student's attempt to check in was successful.
13. The system shall allow the school supervisor to view their students' site visits

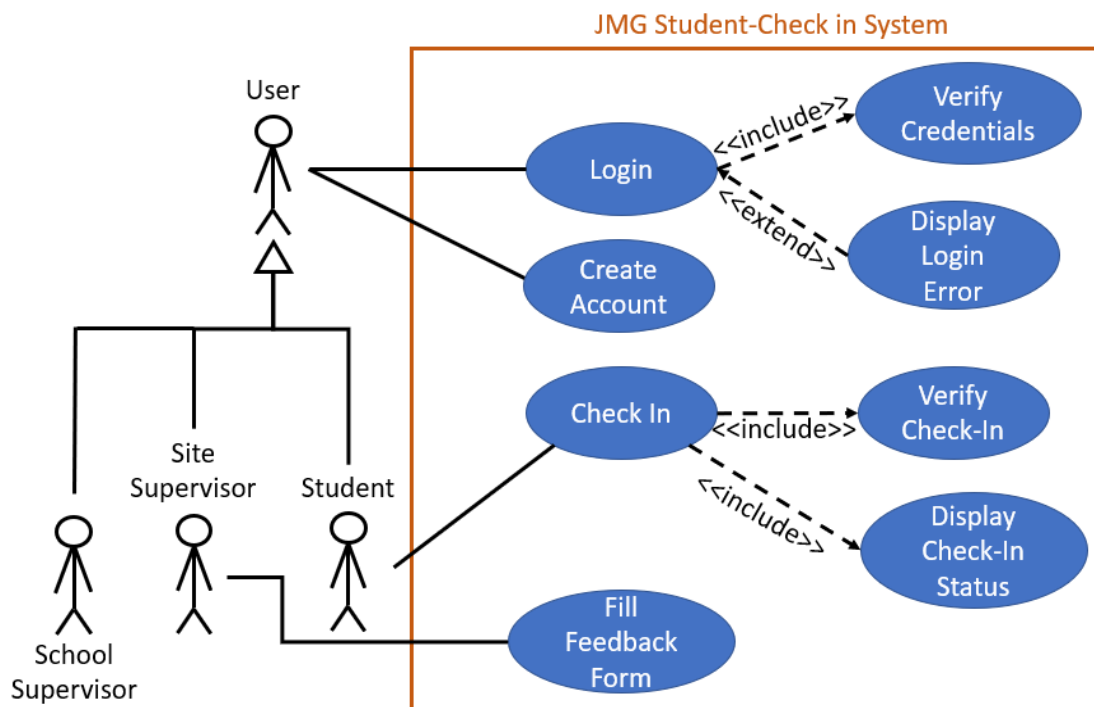


Figure B.II.a: use case diagram for use cases 1 - 3, and 6 - 9

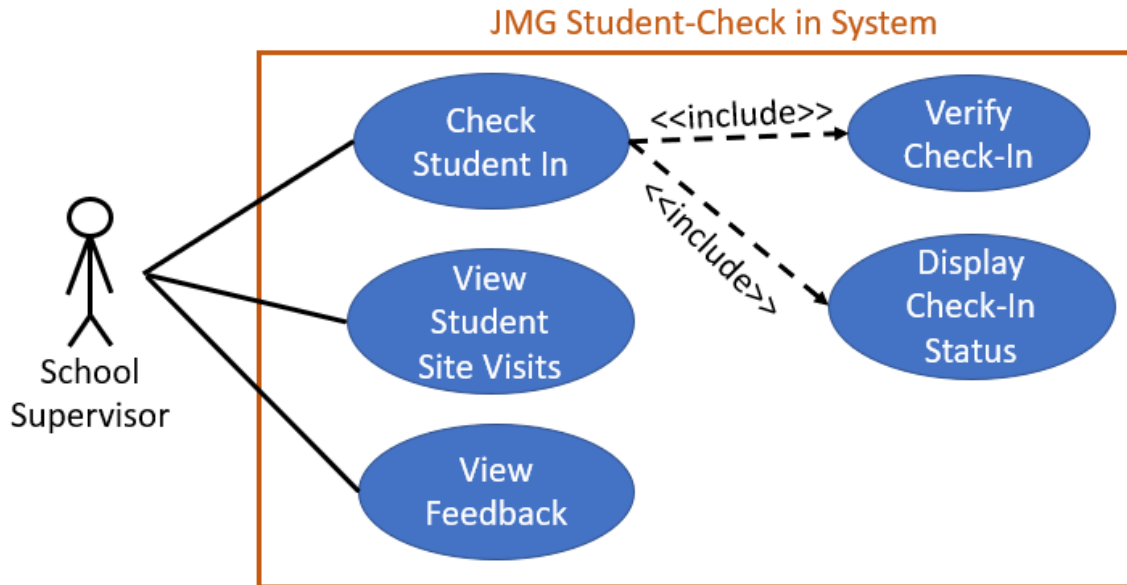


Figure B.II.b: use case diagram for use cases 10 - 13

Based on the functional requirements, most use cases are described using use case specifications.

Number	1	
Name	Log in	
Summary	User logs in by entering a credential combination	
Priority	5	
Preconditions	N/A	
Postconditions	User is viewing main check-in screen	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	User opens check-in application website	
Main Scenario	Step	Action
	1	System displays login screen
	2	User enters a credential combination
	3	System verifies the credential combination
Extensions	Step	Branching Action
	3a	<password invalid>: <system displays small text box notifying user that login failed>
Open Issues	N/A	

Number	2	
Name	Check in	
Summary	Student checks in the site	
Priority	5	
Preconditions	User successfully logged in	
Postconditions		
Primary Actor	Student	
Secondary Actors	N/A	
Trigger	N/A	
Main Scenario	Step	Action
	1	User clicks “check-in” button
	2	System verifies check-in
	3	System displays text box notifying user that check-in was successful
Extensions	Step	Branching Action
	2a	<current time outside of site visit time>: <system displays text box notifying user that check-in failed>
Open Issues	Not sure how to approach unexpected check-ins	

Number	3	
Name	Fill Feedback Form	
Summary	Site supervisor fills out form assessing student performance for session	
Priority	4	
Preconditions	Student is checked in to site, site supervisor has received notification	
Postconditions	Submitted form should be visible to school supervisor	
Primary Actor	Site supervisor	
Secondary Actors		
Trigger	Site supervisor opens feedback form	
Main Scenario	Step	Action
	1	Site supervisor fills out feedback form (text box, checkboxes, etc.)
	2	Site supervisor clicks submit button
	3	system displays message notifying site supervisor that the form was submitted successfully
Extensions	Step	Branching Action
	3a	<form not completed>: <system reloads form highlighting missing information>
Open Issues	NA	

Number	4	
Name	Check Student In	
Summary	School supervisor checks student in when student forgot to check in	
Priority	4	
Preconditions	Student has not checked in to site	
Postconditions	Student is fully checked in to site	
Primary Actor	School Supervisor	
Secondary Actors		
Trigger	School supervisor notified that student was at site-by-site supervisor	
Main Scenario	Step	Action
	1	School supervisor open's student site visit information
	2	School supervisor check's student in
	3	System displays text box notifying school supervisor that student was successfully checked in
Extensions	Step	Branching Action
	3a	<check in fails>: <system displays text box notifying school supervisor that attempt to check in has failed>
Open Issues	NA	

Number	5	
Name	View Student Site Visits	
Summary	School supervisor views all of the site visits of a particular student	
Priority	3	
Preconditions	School supervisor is logged in	
Postconditions	NA	
Primary Actor	School Supervisor	
Secondary Actors	NA	
Trigger	NA	
Main Scenario	Step	
	1	School supervisor clicks on certain student they want info on
	2	System displays options
	3	School supervisor clicks on "View check-in history"
	4	System loads page showing student's check-in history
Open Issues	NA	

Number	6	
Name	View Feedback	
Summary	School supervisor views all performance assessments of student	
Priority	5	
Preconditions	School supervisor is logged in	
Postconditions	NA	
Primary Actor	School Supervisor	
Secondary Actors	NA	
Trigger	NA	
Main Scenario	Step	
	1	School supervisor clicks on certain student they want info on
	2	System displays options
	3	School supervisor clicks on “View performance history”
	4	System loads page showing student’s performance assessments
Open Issues	NA	

Number	7	
Name	Create Account	
Summary	New user creates a new account	
Priority	3	
Preconditions	User is new	
Postconditions	User is now registered in system	
Primary Actor	User	
Secondary Actors	NA	
Trigger	NA	
Main Scenario	Step	
	1	User clicks on create account button
	2	System displays text fields for username, password, and email
	3	User enters username, password, and email
	4	User clicks “create account” button
	5	System verifies information
	6	System sends confirmation email to user
Extensions	Step	Branching Action
	6a	<username/password/email already in system> <system displays message telling user that this username/email/password is already in use>
Open Issues	NA	

We envision the testing of these requirements through a demonstration of the application and displaying each piece of functionality to ensure that the requirements are met. A fake account for a student, school supervisor, and site supervisor will be made for the sake of the demonstration. First a login attempt will be made with an incorrect username and password, followed by the correct username and password. The student account dashboard will be shown, with demonstrations of successfully and unsuccessfully checking in to a site. The site supervisor account dashboard will be shown, with a demonstration of the student assessment form including premature submission. The school supervisor dashboard will be shown, demonstrating their ability to see all their students check ins, as well as the assessments for each session.

III. Non-Functional Requirements

Based on the proposals and proposal review meeting with the client. These non-functional requirements were developed to define the basic performance of the system.

1. The system shall be able to handle 1000 users without impacting its performance. (Lowest Priority)
2. A data request or transmission to the Salesforce system shall take no longer than 10 seconds. (Low Priority)
3. The system should respond to any user request within at least 10 seconds. (High Priority)
4. The system shall perform any database query within at least 10 seconds. (Medium Priority)
5. The system shall follow FERPA guidelines. (Highest Priority)
6. Any new web page must be loaded within 10 seconds of the user initiating it. (High priority)
7. The application should adapt to any size screen including mobile. (Highest Priority)
8. The application should load within at least 10 seconds when there are less than 1000 concurrent users. (High Priority)
9. The application should load within at least 20 seconds when there are more than 1000 concurrent users. (Lowest Priority)
10. The system shall record any failed login attempts. (Low Priority)

IV. User Interface

See “*User Interface Design Document*” for JMG Student Check-in Application.

V. Deliverables

The software should all be available to the clients via the shared GitHub repository including the source code, executables, as well as additional documentations. The source code will include:

- JavaScript, HTML, and CSS files for front end and user interface
- Either Python/Ruby scripts to build backend from scratch
- Use of Budibase as a potential low-code platform

For documentation, the repository will eventually have:

- System Requirements Specification (SRS) (October 25th)
- System Design Document (SDD) (November 10th)
- User Interface Design Document (UIDD) (November 29th)
- Critical Design Review Document (CDRD) (December 13th)
- Code Inspection Report (CIR) (2nd Semester)
- User Manual (UM) (2nd Semester)
- Administrators Manual (AM) (2nd Semester)
- Final Project Report (FPR) (2nd Semester)
- Biweekly Status Reports

There will also be hard copies made of these documents.

Some other documentation that may be included are:

- Sequence Diagrams
- Use Case Models
- Testing Plans (Unit, Regression, Acceptance, Integration, etc.)

VI. Open Issues

These issues will be addressed later in the development process and are as follows:

- Process of determining when a student can no longer check themselves in, especially during unexpected site visits.

Appendix C: System Design Document

JMG Student Site Check-in Application

System Design Document



Client

Lanet Anthony, Samantha Brink, JMG

Developer



Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
November 10, 2021
Version 1.0

JMG Student Site Check-In Application
System Design Document

Table of Contents

I.	Introduction	41
	1. Purpose of the Document	41
	2. References	41
II.	System Architecture	42-44
	1. Architectural Design	42-43
	2. Decomposition Description	44
III.	Persistent Data Design	44-47
	1. Database Descriptions	45-47
	2. File Descriptions	47
IV.	Requirements Matrix	48

I. Introduction

The JMG Student Site Check-In Application is a service that aims to automate the process of JMG students notifying teachers of their attendance at an event outside of school that is counted towards course credit. This is a capstone project for Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse in partial fulfillment of the Computer Science BS degree for the University of Maine.

1. Purpose of the Document

The purpose of this document is to provide descriptions and design specifications for the system to allow for software development to proceed and to have an understanding of what is to be built and how it is expected to be built. This document will generally consist of system architecture design specification, persistent data design, and requirements matrix.

2. References

Budibase. *A beginner's guide to web application development (2021)*. Retrieved November 10, 2021, from [A beginner's guide to web application development \(2021\) \(budibase.com\)](https://budibase.com/blog/a-beginners-guide-to-web-application-development)

Trio. *Web Development in 2021: Everything You Need to Know*. Retrieved November 10, 2021, from [Web App Development in 2021: Everything You Need to Know | Trio Developers](https://trio.dev/blog/web-development-in-2021-everything-you-need-to-know)

React. *A JavaScript library for building user interfaces (2021)*. Retrieved November 10, 2021, from [React – A JavaScript library for building user interfaces \(reactjs.org\)](https://reactjs.org/)

Budibase. *Build internal tools, the easy way (2021)*. Retrieved November 10, 2021, from [Budibase | Build internal tools, the easy way](https://budibase.com/blog/build-internal-tools-the-easy-way)

Django. *The web framework for perfectionists with deadlines*. Retrieved November 10, 2021, from [The web framework for perfectionists with deadlines | Django \(djangoproject.com\)](https://www.djangoproject.com/)

See “*System Requirements Specification*” and “*User Interface Design Document*” for further information.

II. System Architecture

This section provides a general description and a decomposed view of the system architecture. After doing plenty research, the team members conclude that we need three logical layers for the web application: a frontend written in HTML, CSS, and JavaScript with React, a database written in MySQL with Budibase, and a backend written in Python with Django. We choose these development environments because they support great web application integration and are open sourced.

1. Architectural Design

The system architecture contains four major components including the user interface, the system, the database, and the JMG server. User interacts with the user interface, which communicates with the system. The system then updates the database with user input and sends the data to the JMG server.

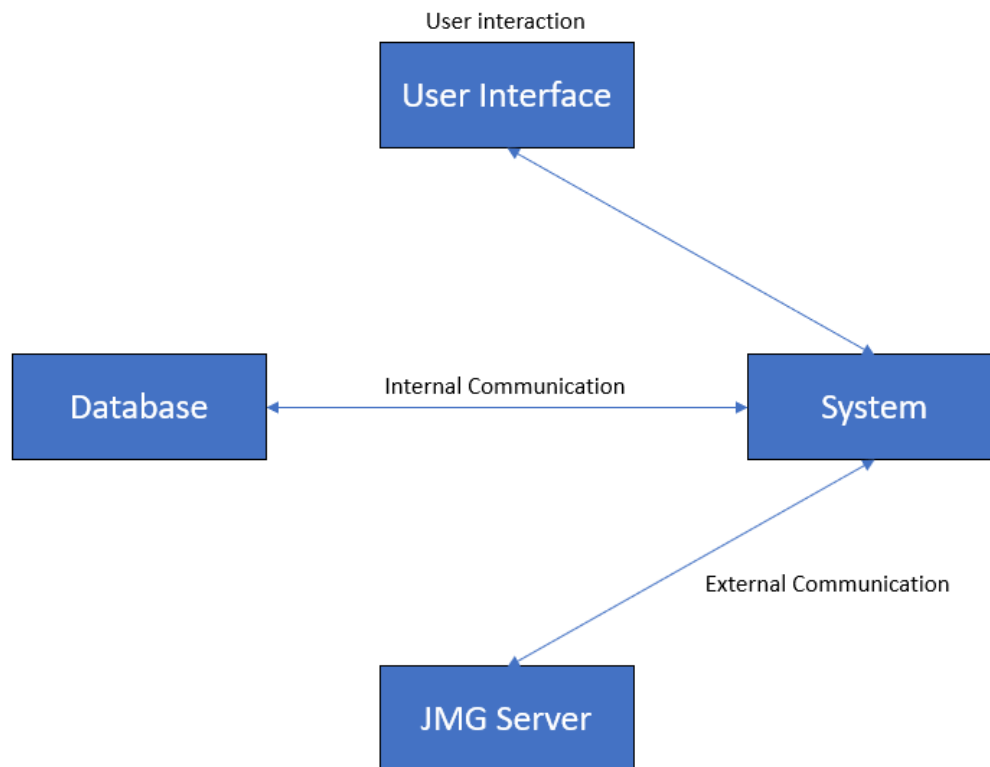


Figure C.VIII.1a: System Architecture Overview – Four major components are shown above, including the user interface, the system, the database, and the JMG server.

The technical architecture contains three logical system components including the frontend, the database, and the backend. The frontend component features a user interface that the user can either interact with on desktop or mobile platforms. Our choice for the frontend platform is React. React is among the most popular JavaScript library for building the user interface featuring declarative development view and encapsulated object component. The database component contains and organizes the user data. Our choice for the database platform is Budibase. Budibase is a user friendly, low-code platform for web application development. The backend component hosts the core functionality of the system. It is responsible for the communication between the user and the main server. Our choice for the backend platform is Django. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

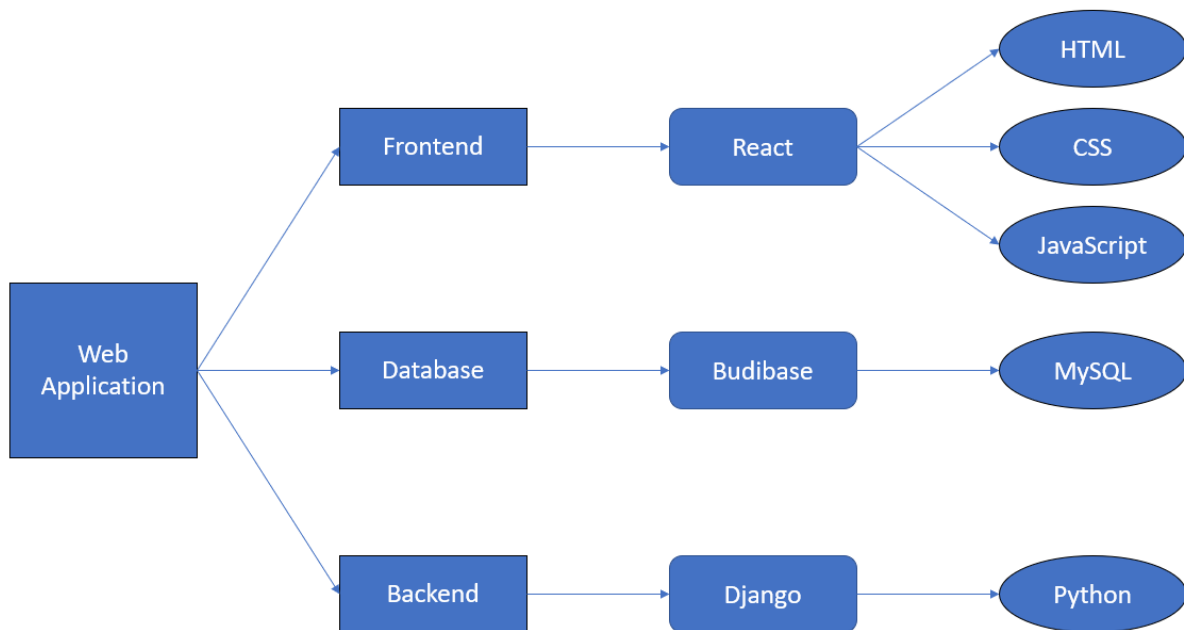


Figure C.VIII.1b: Technical Architecture Overview – The web application is split into three logical components, with the rectangle shape representing components, the squircle shape representing platforms, ad the ellipse shape representing programming languages.

A web application, often referred to as a web app, is an interactive computer program built with web technologies (HTML, CSS, JavaScript), which store (Database, Files) and manipulates data (CRUD), and is used by a team or single user to perform tasks over the internet. The CRUD is a popular acronym and is at the heart of web application development. It stands for Create, Read, Update, and Delete. Web applications are accessed via a web browser such as Google Chrome, Microsoft Edge, Apple Safari, and often involve a login or signup mechanism.

2. Decomposition Description

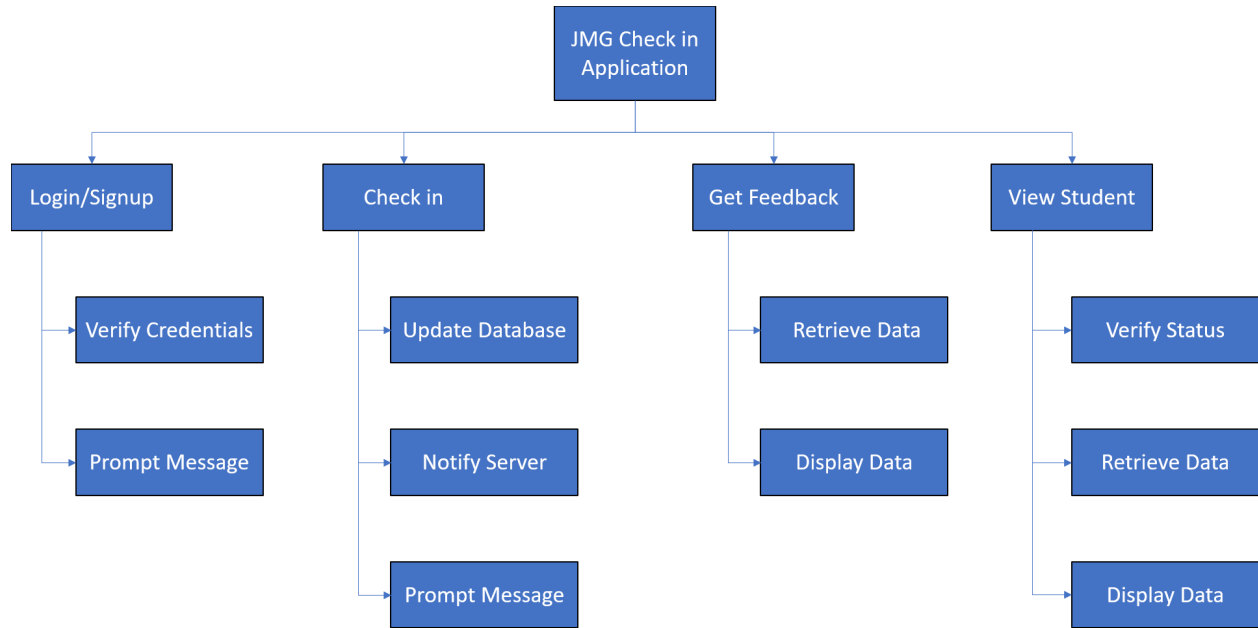


Figure C.VIII.2: Structural Decomposition (Hierarchy) Overview – Four major functions of the web application is shown above, including user login/signup, student check-in, student getting feedback, and supervisor viewing student record.

The Structural Decomposition (Hierarchy) Overview features four major functionalities of the web application. The first functionality is login/signup. The user shall provide a set of credentials, and these credentials will be sent to the JMG server for verification. The second functionality is check-in. A user with student status can click on a button to perform the check-in action with the server. A message will be prompted to the user to inform whether the action was successful. The third functionality is getting feedback. A student user can request feedback data from the server, and the server will provide the student with its information. The last functionality is viewing student. A user with supervisor status can view students' information. This action can only happen when the request's supervisor status is verified.

III. Persistent Data Design

This section will provide descriptions and diagrams of the database structure used in the system and also the file(s) used by the system.

1. Database Description

A relational database will be needed to help maintain records such as account usernames, passwords as well as other requirements such as viewing past check-ins, feedback, etc. To help encapsulate what this will look like, an entity-relationship (ER) model will be used to describe the relationships.

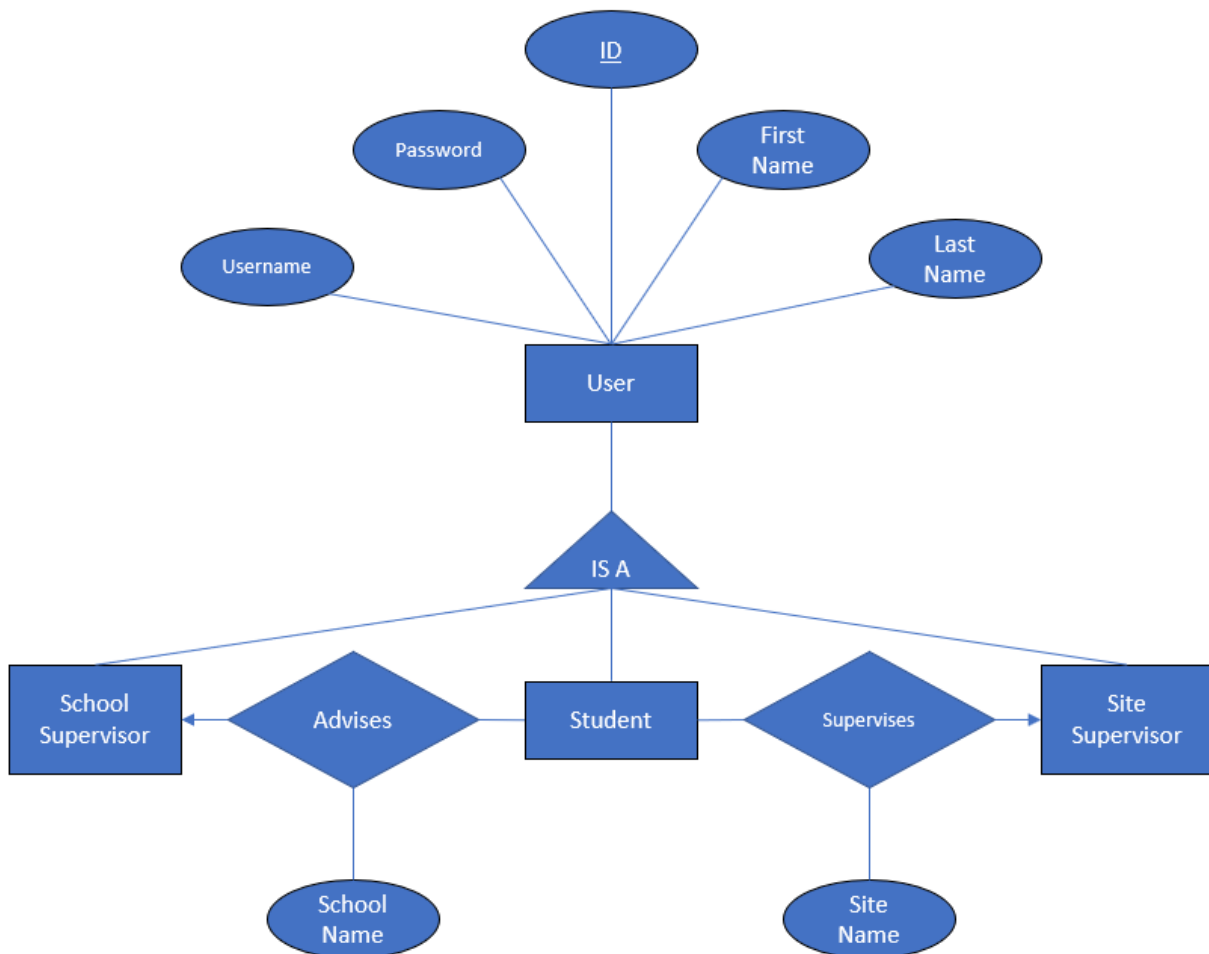


Figure C.IX.1a: ER Model for User Table – Each user will have a username, password, ID number, first name, and last name. The student, school supervisor, and site supervisor are all users linked to each other via the supervises and advises relations.

The first part of the ER model describes what data might be held for each user regardless of their role in the check in process. The ID serves as the key attribute due to its uniqueness for each user. The second part of the ER model divides the different kinds of users in the system. There are two relationship sets that link each of the users. Between the school supervisor and the student, the “advises” relation will describe which students

are being advised by which teachers. There is a one-to-many/many-to-one relationship here as a school supervisor can advise multiple students, but a student is assigned only one school supervisor. The “supervises” relation links the student and site supervisor in which again is a one-to-many/many-to-one relationship in which a student is assigned to only one site supervisor, but a site supervisor can be in charge of multiple students.

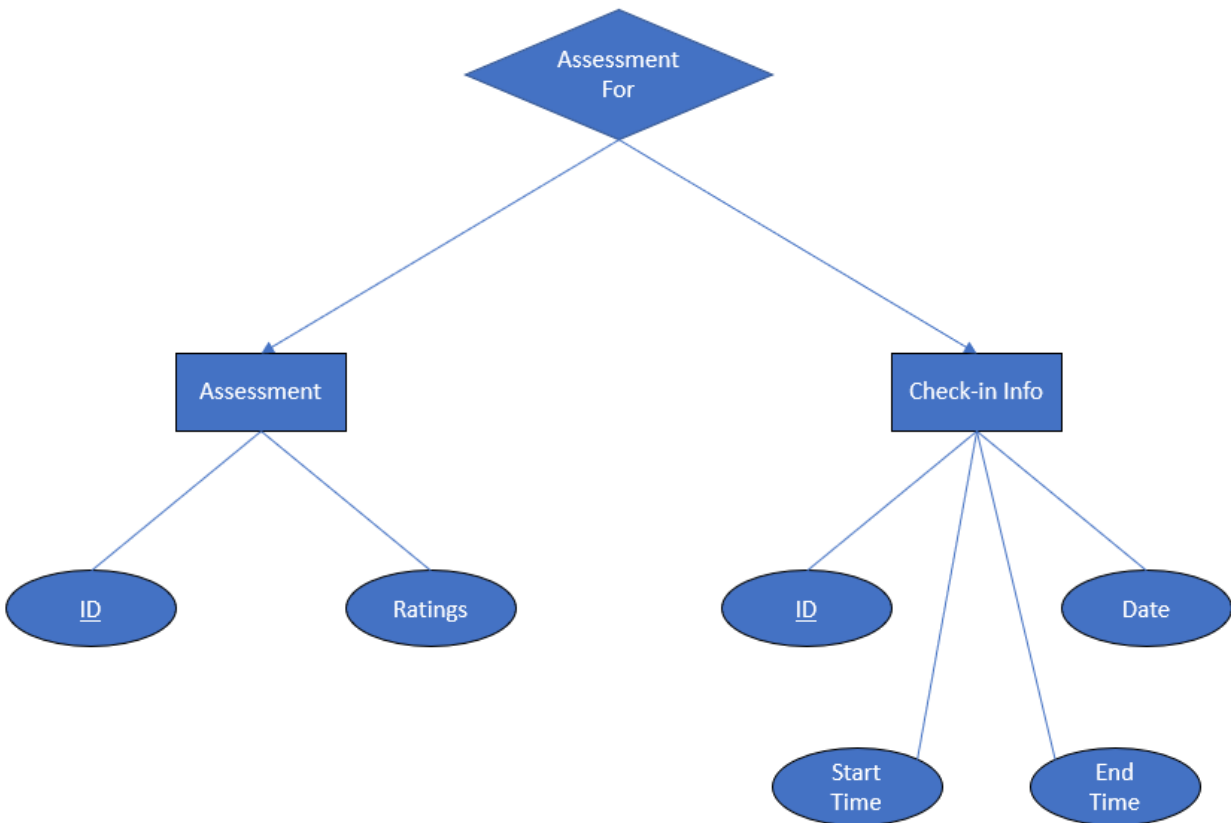


Figure C.IX.1.b: Site Check-in Model, Assessment Table, and Relationship with Check-in – The student is also linked to check-ins, which have a start time, end time, date, and id. Each check in will have an assessment to go along with it.

The “advises” relation has an additional attribute describing the name of the school the student and teacher attend. The “supervises” relation also has an additional attribute describing the name of the site (the company name). The site check-in entity has four attributes including the assessment, date of the check-in, and the start and end time of the session. The student and site check-in entities are linked by the “check-ins” relation, which has a one-to-many/many-to-one relationship where a student has multiple check-ins, but a check-in belongs to one specific student. Each check in has an id, date, start time, and end time. Once again, the ID serves as the primary attribute due to its uniqueness. Since each site visit must have a performance assessment afterward, the

performance assessment has its own table. Each assessment has an ID as well as various fields for ratings that haven't been decided on yet. They will be of data type integer or string depending on the rating scheme. Each assessment is linked with exactly one check-in (hence the curved arrow). And the check-in has at most one assessment.

Database Schema (with data types):

```
User(id(int), username(string), password(string), fname(string),  
lname(string), role(string))  
Advises(tid(int), sid(int), school(string))  
Supervises(supid(int), stid(int), sname(string))  
Check-in(id(int), date(int), stime(string), etime(string))  
Check-ins(sid(int), cid(int))  
Assessment(id(int), ...)  
AssessmentFor(aid(int), cid(int))
```

2. File Description

The system does not require the use of any additional files, although this may change as the development progresses.

IV. Requirements Matrix

This section will provide a table that is designed to show which components satisfy each of the functional requirements referenced in the SRS document.

	Login	Database	Rest API	Privacy	Check-in	GUI	Admin	Feedback
Create Account	X							
Login	X	X						
Verify Credentials	X	X						
Send Data to API		X	X					
Get Data From API		X	X					
Data Privacy				X				
Check-in					X			
Notify Student If Checked-in					X	X		
Notify Supervisor of Student Check-in					X	X	X	
Feedback Form							X	X
View Feedback							X	X
Supervisor Check-in					X		X	
Supervisor Check-in					X	X	X	
View Student Visits						X	X	

Appendix D: User Interface Design Document

JMG Student Site Check-in Application

User Interface Design Document



Client

Lanet Anthony, Samantha Brink, JMG

Developer



Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
November 29, 2021
Version 1.0



JMG Student Site Check-In Application
User Interface Design Document

Table of Contents

I.	Introduction	51
	1. Purpose of the Document	51
	2. References	51
II.	User Interface Standards	51-54
III.	User Interface Walkthrough	55-59
IV.	Data Validation	60-61

I. Introduction

The JMG Student Site Check-In Application is a service that aims to automate the process of JMG students notifying teachers of their attendance at an event outside of school that is counted towards course credit. This is a capstone project for Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse in partial fulfillment of the Computer Science BS degree for the University of Maine.

1. Purpose of the Document

The purpose of this document is to process the product requirements into a more detailed format and capture the details of the software user interface into a written document. The content within this document will include the user interface design standards within the system, a walkthrough of the user interface, a description of the data items that will be used in the system, and any report formats used if any.

2. References

Framer. *A Free Prototyping Tool for Teams* (2021). Retrieved November 29, 2021, from [Framer: A Free Prototyping Tool for Teams](#)

See “*System Requirements Specification*” and “*System Design Document*” for further information.

II. User Interface Standards

This section provides a general graphical user interface mockup. The JMG Student Check-in is directly split into 5 sections: the home page, the curriculum page, the assessment page, the profile page, and the credential page. Other webpages (including variations for the student user, the instructor user, and the administrator) are discussed in Section III. Figure II.1 shows the home page view. Some general information of the website is shown to the user, including JMG introduction, FAQs, and contacts. Figure II.2 shows the curriculum page view. Available courses and their schedules will be listed here. Figure II.3 shows the assessment page view. The user can access to their course assessment form here. Figure II.4 shows the profile page. The user can view and edit its full name, school name, and ID here. Figure II.5 shows the credential page view. A user must login or signup before using features on the website. The user can also find back its credentials using email verification.

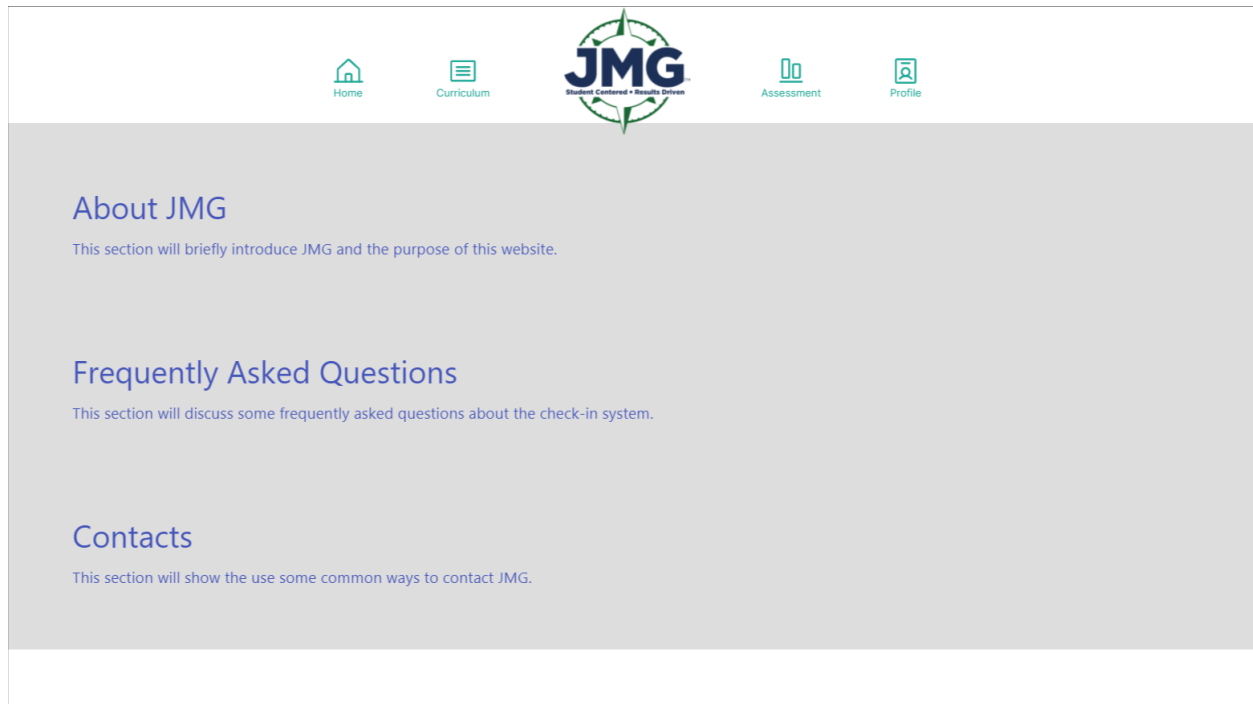


Figure D.II.1: The home page view of the system. There are three sections, including “About JMG”, “Frequently Asked Questions”, and “Contacts”.

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 AM	COS 397		COS 397		
09:00 AM					
10:00 AM		COS 490		COS 490	
11:00 AM	COS 331		COS 331		COS 331
12:00 PM					
01:00 PM	COS 497		COS 497		

Figure D.II.2: The curriculum page view. Available courses and their schedules will be listed here.

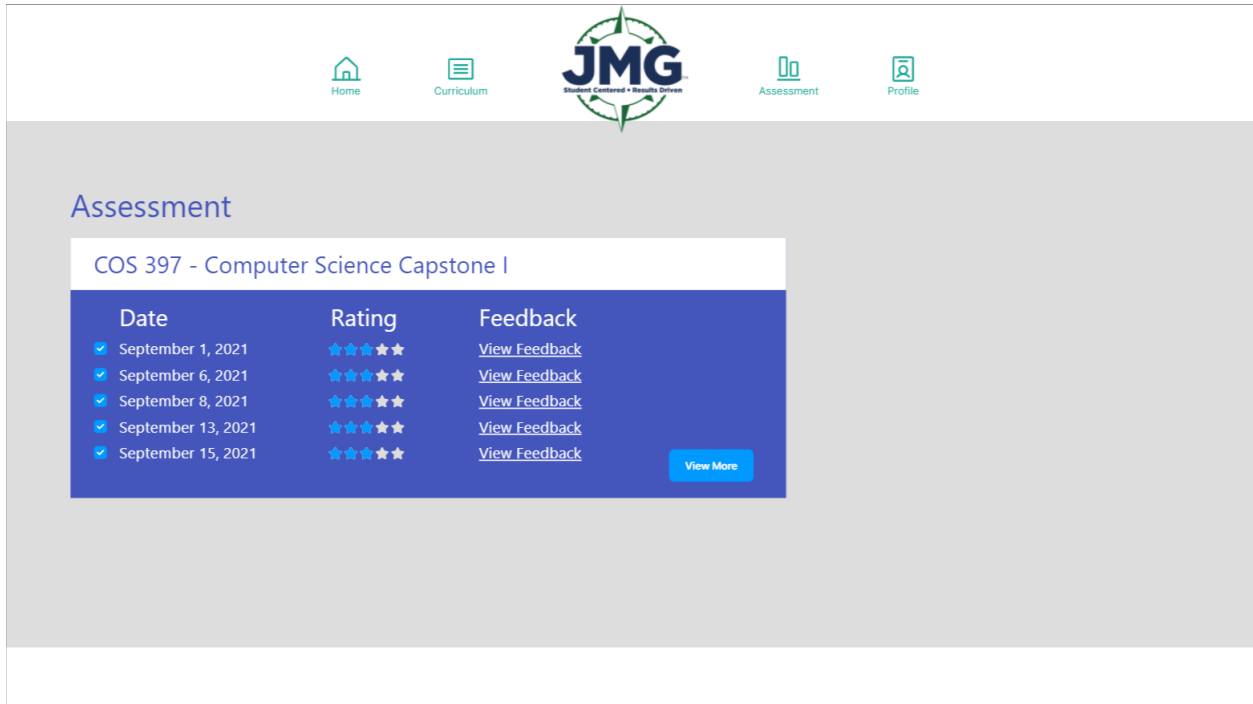


Figure D.II.3: The assessment page view. The user can access to their course assessment form here.

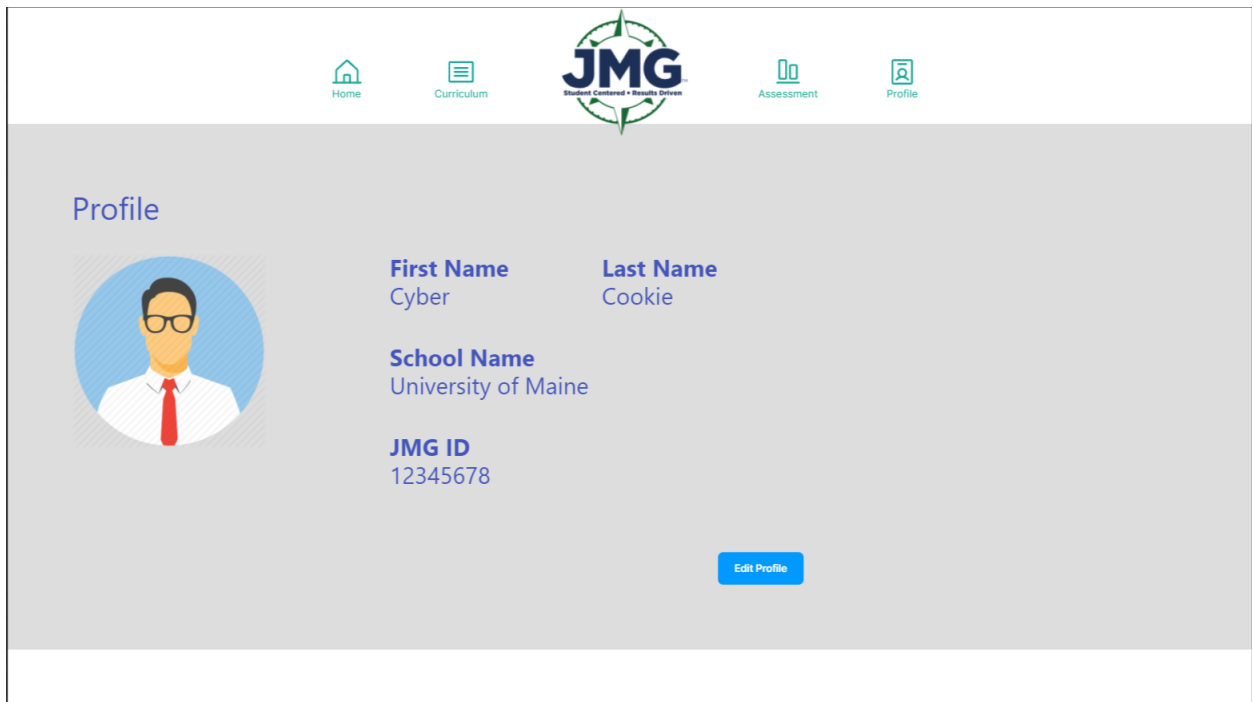


Figure D.II.4: The profile page. The user can view and edit its full name, school name, and ID here.



Figure D.II.5: The credential page view of the system. A user must login or signup before using other features on the website.

III. User Interface Walkthrough

Navigation Diagram

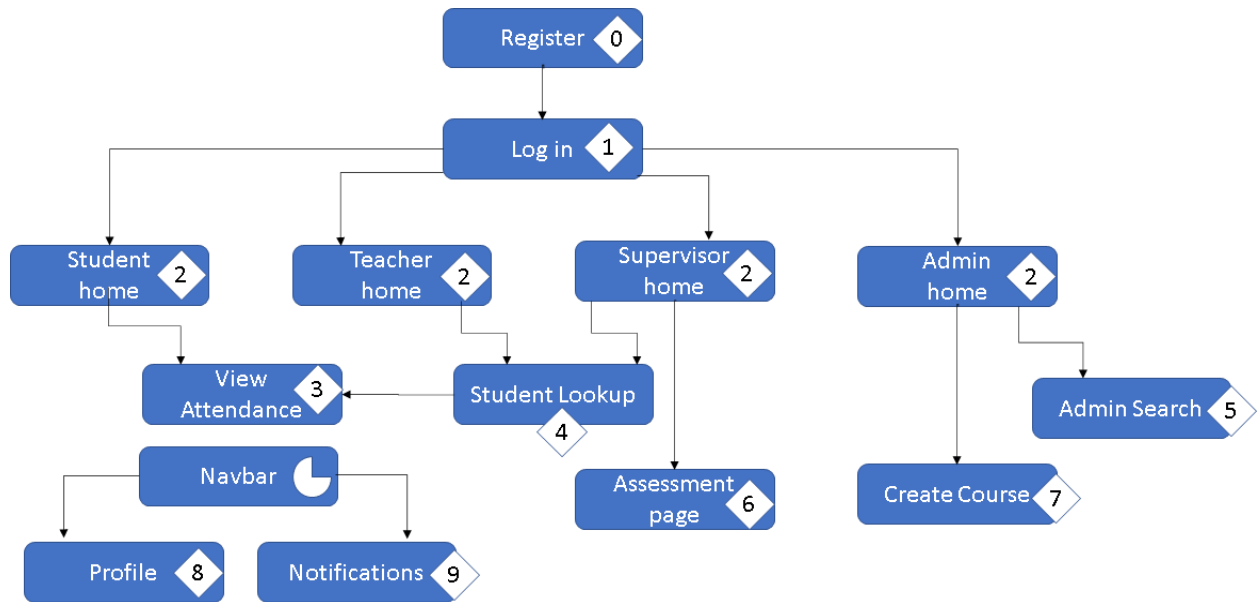


Figure D.III.1: Site Navigation Flow Chart

This diagram shows how the pages currently designed relate to one another and how to navigate to them. At this point in time almost all pages are directly accessible from the homepage, with the exception of “view attendance” which requires a student lookup for teachers and supervisors. Each homepage is relatively similar, with the exception of the buttons available to each user. In general, it should be assumed that the snapshots don’t represent small differences between user roles, but rather represent a structural form universal to each user.

Web Page Views

0

Register

Username

{Username}

Password

Name

First...

Last...

Role

▼ {Role}

Organization

{Organization}

Submit

1

Log in

Username

{Username}

Password

Submit

Figure D.III.2: Registration and Figure D.III.3: Login

Home

7

Profile

2

Home

{Course Title}

Check-In

No Check-In detected

View Attendance

{Course Title}

Check-In

Checked in Today ✓

View Attendance

...

Home

7

Profile

3

Attendance

Average

4.5/5

View

11/16/21

5/5

View

11/16/21

4/5

View

{date}

{rating}

View

{date}

{rating}

View

{date}

{rating}

View

{date}

{rating}

View

{date}

{rating}

←

{Page #}

→

56

Figure D.III.4: Home and Figure D.III.5: Attendance

The image shows two mobile app screens side-by-side. Both have a dark blue header with 'Home', a white circle icon, a blue circle with '7', a bell icon, and a 'Profile' link. The left screen is titled 'Student lookup' with a diamond icon containing the number '4'. It has a 'Profile' button, a search bar, and a list of entries. Each entry shows '{Course}' and '{Name}', followed by 'Assessment' and 'Attendance' buttons. The entries are: 'Lemon Co. Cave Johnson' and 'States Craft Alexander Hamilton'. The right screen is titled 'Admin Search' with a diamond icon containing the number '5'. It has 'Profile' and 'Course' buttons, a search bar, and a list of entries. Each entry shows '{Course}' and '{Name}', followed by 'Delete' and 'Sign In' buttons. The entries are: 'Lemon Co. Cave Johnson' and 'States Craft Alexander Hamilton'. Both screens have a bottom bar with left and right arrows and '{Page #}'.

Figure D.III.6: Student Lookup and Figure D.III.7: Admin Search

The image shows two mobile app screens side-by-side. Both have a dark blue header with 'Home', a white circle icon, a blue circle with '7', a bell icon, and a 'Profile' link. The left screen is titled 'Assessment' with a diamond icon containing the number '6'. It has a '{Course Title}' header, a '{Student} reported:' section with 'Date' (11/16/21) and 'At' (8:37 AM) fields, 'Confirm' and 'Deny' buttons, 'Review Criteria #1' with five circles (four grey, one white), and a 'Free Write' section with the text 'Apprentice seemed a little tired. Overall good job. Keep up the good work.' Below this is 'Overall 4.2/5' and a 'Submit' button. The right screen is titled 'Course Creation' with a diamond icon containing the number '7'. It has a 'Name' field with '{course title}' and a 'Generated ID: J4K66OPX3S19' label, followed by a 'Create Course' button.

Figure D.III.8: Assessment and Figure D.III.9: Course Creation

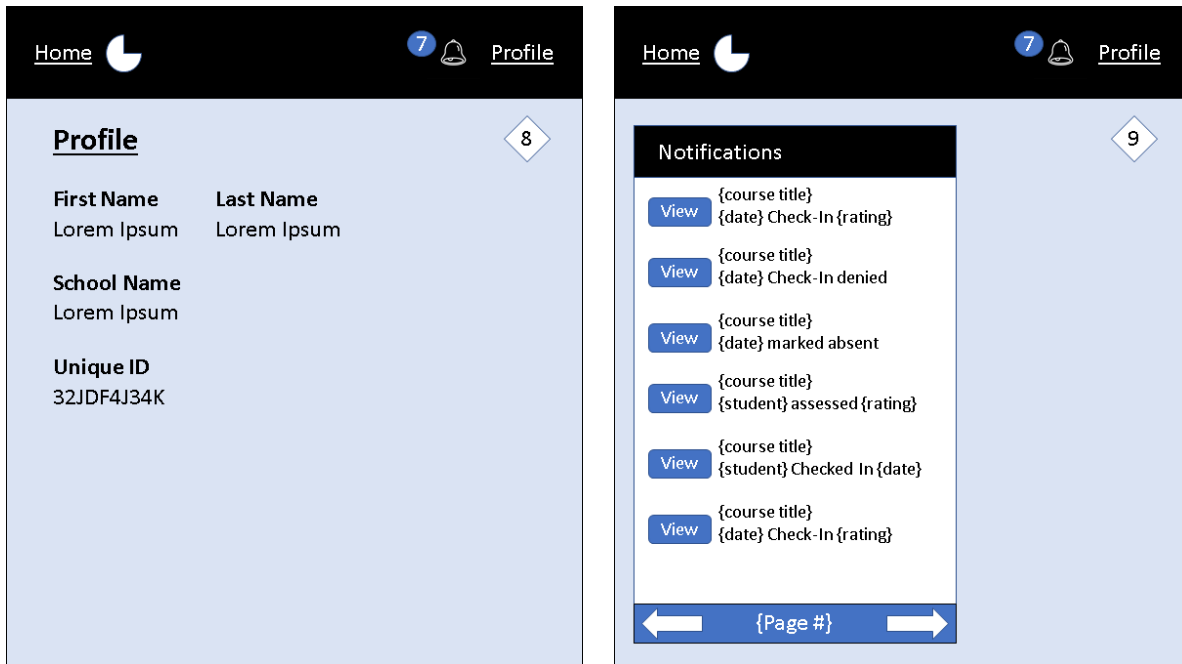


Figure D.III.10: Profile and Figure D.III.11: Notifications

Figure Overview

3.2. The registration page consists of a series of user input, including username, password, full name, role, and organization(s). Upon pressing submit this information is sent to our servers and a new profile is created.

3.3. The Login page takes a username and password and logs you into the associated account if a match is found in our database.

3.4. The homepage is the basis of our navigation tree and provides course specific subpages and buttons for teachers, supervisors, and admins to find other users (or courses). The check in button is student specific and allows a student to mark themselves as present, the current time will be automatically provided.

3.5. The attendance page is viewable by a student and any other role that is associated with that student and displays an average assessment score, as well as allowing for all individual assessments to be viewed.

3.6. The student lookup allows for supervisors and teachers to search through the students under them to find one individual. The buttons available to each role differ, but a supervisor can write an assessment, while both can view the students' attendance here.

3.7. The Admin search allows for toggling between courses and users. An admin can delete any user or course, while also wielding the power to log in as any individual to view their profile.

3.8. The Assessment page consists of an attendance submission which can be confirmed or denied (if none then a button will be available to mark absent), a set of rating criteria for the student, and a free write portion. The overall grade is (currently) calculated automatically.

3.9. Course creation is in the hands of an admin account, when a new course is created a random ID is generated.

3.10. The profile page contains basic information that our database has on a user, such as their name, organization, and ID.

3.11. The notifications page is the same for all roles, but the notifications that each role can receive are very different. In most cases the view button will take you to your (or a student's) assessment. Students and teachers are alerted of their rating, while supervisors are alerted of check-in submissions.

navbar (¾ circle): Appears on every page after log in, the navbar is treated differently and separated on the navigation diagram. It has a button to link back to the home page, the notifications page, and the user profile.

IV. Data Validation

Each user, depending on what type they are, will be able to enter in information that will be stored in the database. For example, when a user creates an account, they will enter their first and last name. Some of this data must be entered in a specific format as described by the table below.

Item	Data Type	Limits	Allowable Format
Login: Username	String	5 - 20 characters long	No specific format
Login: Password	String	No special characters other than '\$', '!', '#', '& 5 - 20 characters long	Contains at least one letter, number, and special character.
Create Account: First Name	String	1 - 30 characters	No specific format
Create Account: Last Name	String	1 - 30 characters	No specific format
Create Account: School	String	5 - 80 characters	Any format
Create Account: User Role	String	NA	“teacher”, “site supervisor” or “student” Selected from drop down
Create Account: Company Name	String	1 - 100 characters	No specific format
Check In: Date	String	Max Day: 31 Max Year: 9999 Max Month: 12	“DD/MM/YYYY”

Item	Data Type	Limits	Allowable Format
Check In: Begin Time	String	Max Hour: 12 Max Minute: 59	“HH:MM (meridiam)” or “HH:MM(meridiam)”
Check In: End Time	String	Max Hour: 12 Max Minute: 59	“HH:MM (meridiam)” or “HH:MM(meridiam)”
Assessment: Effort	Integer	1 - 5	User clicks on correct checkbox
Assessment: Professionalism	Integer	1 - 5	User clicks on correct checkbox
Assessment: Attentiveness	Integer	1 - 5	User clicks on correct checkbox
Assessment: Overall Performance	Integer	1 - 5	User clicks on correct checkbox
Assessment: Additional Comments	String	400 characters	No specific format

Table D.IIV.1 - Data Validation - These are all of the different kinds of data that can be entered by a user of the application. Each has a data type, and some have specific limits and formats.

Appendix E: Code Inspection Review

JMG Student Site Check-in Application

Code Inspection Report

Client



JMG

Lanet Anthony, Samantha Brink

Developer



Cyber Cookie

Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine

March 9, 2022

Version 1.1



JMG Student Site Check-in Application Code Inspection Report

Table of Contents

1. Introduction	64-65
1.1 Purpose of the Document	64
1.2 Design Conventions	64
1.3 Defect Checklist	65
1.4 References	65
2. Design Inspection Process	66-67
2.1 Description	66
2.2 Impressions of the Process	66-67
2.3 Inspection Meetings	67
3. Inspected Modules	68-69
4. Known Defects	70-71
Appendix A: Design Conventions	72

1. Introduction

The JMG Student Site Check-In Application is a web service that aims to automate the process of JMG students notifying teachers of their attendance at events outside of school counted towards course credit. This is a capstone project for the Cyber Cookie team, which includes Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse, in partial fulfillment of the Computer Science BS degree for the University of Maine. The deliverable of this project practices the development cycle of industrial standards. During the previous semester, the Cyber Cookie team had established system requirements, architecture definitions, and user interface designs. The goal for the team this semester is to implement and integrate these modules into the desired functional system with supportive documentation.

1.1 Purpose of the Document

The purpose of this document is to demonstrate our process by reviewing system design at the modular and atomic level and showcase the results of our inspections. More specifically, this document is meant to elaborate the design philosophy and discuss system defects found during inspection or may appeal in the future.

1.2 Design Conventions

Given the nature of the project, after careful discussions, the team decided that the Budibase would serve perfectly to be the platform the system should be built on. The development process was not programming intensive. Most of the design process was done by using the GUI system of the Budibase. Being the core of internet scripting, we also used JavaScript to regulate behavior between website interactions. For the purpose of efficient and effective development, we have established our own design conventions which are mentioned in the Appendix A.

1.3 Deflect Checklist

Defect #	Defect	Critical	Major	Minor
1	Logic Errors		X	
2	Security Oversights	X		
3	Automation Errors		X	
4	User Interface Defects			X
5	Performance Issues			X
6	Incorrect Database Relations		X	
7	Computational Errors		X	
8	Improper use of Budibase Features			X
9	Boundary Conditions		X	
10	Ambiguous Design			X
11	Missing Design		X	
12	Navigational Errors		X	
13	Handlebar Errors		X	
14	JavaScript Conventions			X
15	JavaScript Errors		X	
16	Permission/Access Control Issues	X		

Table 1.3 - Defect Checklist: These are the potential errors/defects that we ended up looking for during inspection.

1.4 References

See *System Requirements Specification* for information regarding system requirements.

See *System Design Document* for information regarding system design.

See *User Interface Design Document* for information regarding user interface design.

2. Design Inspection Process

2.1 Description

As our team is using a platform that replaces most mandatory coding, we inspected and made an analysis of individual units and components of the system including the user interface, the database, and any of the automations that we have set up using the provided builder. As a result, we reviewed the different uses of the various tools that the builder provides and whether they were implemented correctly. The user interface can be effectively described as a setup of individual “screens”, each of which contain functionality for the given requirements that were laid out in our *System Requirements Specification* document. These screens contain components that are used to interact with the databases. In the settings for these components, conditionals, filtering, navigation, and automation triggers can be set up. There was some custom JavaScript involved in the UI design, which was also inspected for following proper conventions.

The process involved the author of a specific component or module doing a walkthrough of how it was built. One member recorded all observations and defects discovered during these walkthroughs while the remaining members served as something similar to a product owner in a scrum environment. They would attempt to point out as many defects as possible both in design and functionality. Unlike the product owner however, they would also be on the lookout for more currently present and potential technical issues. After a walkthrough of their build, the author then ran a simulation to demonstrate the functionality of their design more clearly and search for more defects through things such as various I/O and navigation errors.

2.2 Impressions of the Process

The inspection process was greatly modified due to the much smaller amount of code required to be written and smaller number of people involved. The team believe that the walkthrough process was the most useful because people are very good at explaining what they wrote themselves and, in addition, can listen to themselves about how they went about doing a certain component. A lot of times the author themselves would pick up on certain potential defects or already present defects as they are doing the walkthrough. However, perhaps we should have run the simulations more than the few times we did run just to explore all the possible inputs, outputs, state changes, and table updates and possibly detect more errors.

Among all modules the team had explored, we were most positively impressed by the basic portal navigation. Since the system involves users with different levels of access priority, for example, an administrator is able to access all pages (all page tabs are visible to an administrator), some users are not able to view certain pages. The button on each web page also worked as intended. We managed to sign into the site as an administrator and were able to review students' check-in conditions.

Aside from the overall competent system, some minor defects of the system popped out during the inspection. For example, the course page typically displays courses in two separated columns. When the number of courses is even, the page looks fine, but when that number is odd, the course on the last row would occupy the entire row. This is not a functional defect, but an aesthetic one. The team plans to take care of these design issues over the coming break.

2.3 Inspection Meetings

We held two formal design inspections:

March 8th, 2022 (virtually) from 6:00 PM - 7:30 PM:

- **Participants:** Elijah Caret (Author and Moderator), Michael Ferris (Recorder and Observer/Product Owner), Xingzhou Luo (Observer/Product Owner), Spencer Morse (Recorder and Observer/Product Owner)
- **Covered Components:** User identification, enrolled ELOs/courses display, check-in, check-in verification, and basic portal navigation

March 9th, 2022 (virtually) from 6:00 PM - 7:30 PM:

- **Participants:** Elijah Caret (Moderator, Recorder, and Observer/Product Owner and), Michael Ferris (Author and Observer/Product Owner), Xingzhou Luo (Observer/Product Owner), Spencer Morse (Observer/Product Owner)
- **Covered Components:** Homepage, course page, course adding page

3. Inspected Modules

Finished Modules:

- **Check-in:** this module allows the student to perform check-in action.
- **Check-in Verification:** this module allows the instructor to confirm a student's check-in action.
- **Portal UI:** this module provides the user with a graphical interface that interacts with other pages on the site.
- **Course/ELO Adding:** this module allows the student to add courses/ELOs to their schedule.
- **Course Enrollment:** this module allows the instructor to add students to a course.
- **User Database:** this module stores the data of the site.
- **Check-ins Database:** this module stores the check-in records.

Unfinished Modules:

- **Salesforce Integration:** this module enables communication between the JMG app to the Salesforce site.
- **Performance Assessment:** this module evaluates the robustness of the system.

Some of the current system implementations differ moderately from the original plan proposed in the *System Design Document*. In the Architectural Design section, we originally planned the system architecture as the Figure 3.1 showing below.

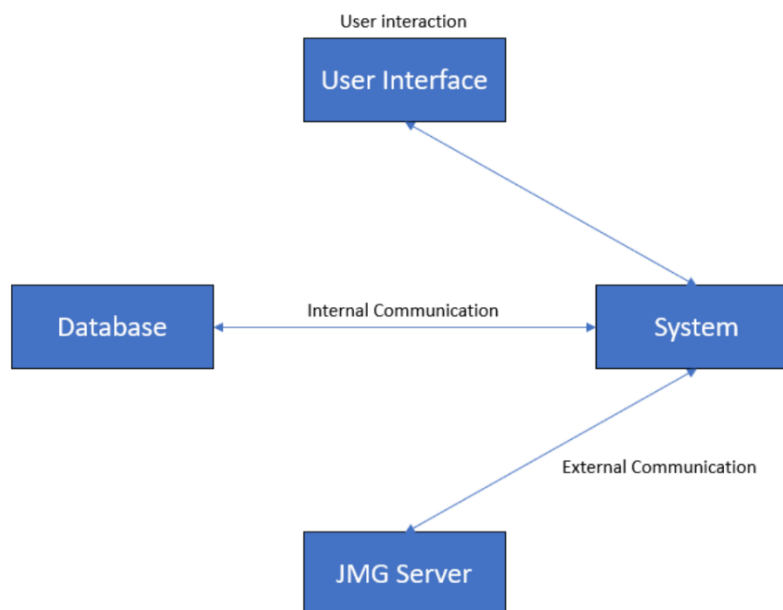


Figure 3.1 - System Architecture Overview.

Note that in the overview, there are four major components. Now, since we adopt the Budibase as our development platform, we can use its internal database module. We are also going to implement the Salesforce integration, which allows communication of the system with the JMG server via the REST API. We have also made changes to the Technical Architecture section shown in the Figure 3.2 below.

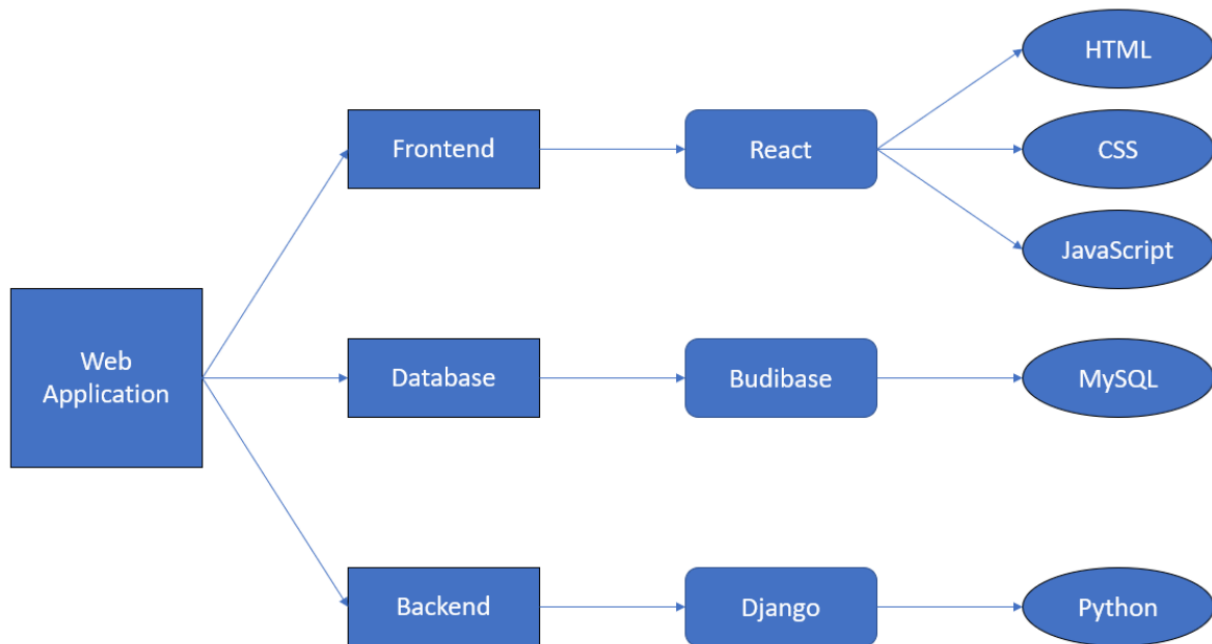


Figure 3.2 - Technical Architecture Overview.

In the original plan, we split the web application into three major logical separations. The robustness of the Budibase software allowed us to do all these works on a singular platform. The Budibase includes a full database system (we could also integrate an external database of our liking), a graphical action interface, and programming supplementation. Thus, the only programming languages we must work with are reduced to JavaScript and CSS, making the project organized and straightforward. Other design philosophies saw only minor changes, for example, we used temporary mockup pictures for our UI design overview, and it is going to be different in the actual implementation.

4. Known Defects

Description of Defect	Module Defect Found In	Defect Categories
When accessing the drop-down tab to choose the company/ELO you are checking into, nothing shows up	Check-In Form Fill out	Correctness, Handlebar Errors, Database Failures, Communication Errors, UI Defect, Missing Design
Users are identified by name instead of identification number	User Database	Database Errors, Security Oversights, Relational Database Errors
Date selection is not bounded, a user can enter any date and ultimately session durations that are nonsensical	Check-In Form Fill out	Boundary Conditions
Submit button is positioned very poorly on form	Check-In Form Fill out	UI Defect, Missing Design
Can't go back to homepage after verifying check-in	Check-In Verification	Navigational Errors
Relationship between Student users and the ELOs doesn't allow for much flexibility when accessing data between the two	User Database	Incorrect Database Relations, Improper Use of Budibase features
Student can access the check-in verification page, which is only meant for supervisors or admins	Homepage UI	UI Defect, Conditional Errors, Permissions/Access Control Issues
Check-in form takes more than 5 seconds to load	Check-In Form Fill out	Performance Issues

Table 4.1 - All Defects Detected: Included are the module they belong under as well as the related defect categories

Description of Defect	Module Defect Found In	Defect Categories
The last course card is formatted to stretch to two card slots if an odd number of courses are taken.	Course formatting	UI Defect.
The view course page inconsistently updates page contents since last visited course, causing the wrong page to load.	Course page	Navigational Errors, UI Defect, Communication Errors.
Add course button was added to the UI, but has no functionality.	Add course button	Missing Design, Navigational Errors.
Not all components are given unique names or even changed from the default value.	Naming conventions	Ambiguous Design.
Not all relational databases are set up properly, some are set as one - many instead of many - many.	Relational database	Logic Errors, Incorrect Database Design.

Table 4.2 - All Defects Detected (continued from 4.1): Included are the module they belong under as well as the related defect categories

Appendix A: Design Conventions

In Budibase, most components are given labels for identification. As a result, we decided that labeling these components properly would help ensure other members on what those components do and how they are related to the app itself. We use a string that starts with one or multiple words to indicate the primary entity that triggers the action and ends with an object to indicate the primary action that is triggered. Note that all words are started with an uppercase letter and separated by one space. Some examples are shown as follows.

- To indicate what a component is or its functionality. For example, if one makes a button to let the student to check-in, they would label the component as “Check-in Button”.

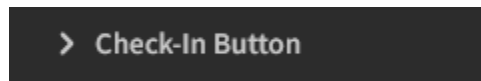


Figure A.1 - Example label of a button: “button” is included in its label.

- To indicate the table that the component is drawing data from. Data provider components for example provide access to data, and hence should indicate the source in their label.

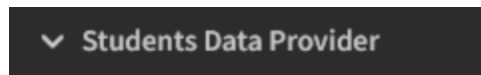


Figure A.2 - Data Provider label: “Students” indicates the table it is drawing data from.

One other convention that must be followed is based on creating a table. In that all data in each table must have an auto identification number. This is to help distinguish any duplicates made and help log students who may have the same name.

Appendix F: Administrator Manual

JMG Student Site

Check-in Application

Administrator Manual

Client



JMG

Lanet Anthony, Samantha Brink

Developer



Cyber Cookie

Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine

April 3, 2022

Version 1.1



JMG Student Site Check-in Application Administrator Manual

Table of Contents

	<u>Page</u>
Table of Contents	74
1. Introduction	75
1.1 Purpose of This Document	75
References	75
2. System Overview	76
2.1 Background	76
2.2 Hardware and Software Requirements	76
3. Administrative Procedures	77-86
3.1 Installation	77
3.2 An Important Distinction	77-78
3.3 Routine Tasks	78-84
3.4 Periodic Administration	85-86
3.5 User Support	86
04. Troubleshooting	87
4.1 Dealing with Error Messages and Failures	87
4.2 Known Bugs and Limitations	87

1. Introduction

The JMG Student Site Check-In Application is a web service that aims to automate the process of JMG students notifying teachers of their attendance at events outside of school counted towards course credit. This is a capstone project for the Cyber Cookie team, which includes Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse, in partial fulfillment of the Computer Science BS degree for the University of Maine. The deliverable of this project practices the development cycle of industrial standards. During the previous semester, the Cyber Cookie team had established system requirements, architecture definitions, and user interface designs. The goal for the team this semester is to implement and integrate these modules into the desired functional system with supportive documentation.

1.1 Purpose of This Document

This document is the administrator manual for the JMG Student Site Check-In Application designed by Cyber Cookie. It lays out the responsibilities of the system administrator. These responsibilities include any routine maintenance required, which may occur daily. There are also descriptions of any periodic maintenance required, which may occur every few months. Also included in this document are troubleshooting tips as well as the current status of user support. Any software or hardware requirements for setup are included as well, although not much is required for this system.

References

See *System Requirements Specification* for information regarding system requirements.

See *System Design Document* for information regarding system design.

See *User Interface Design Document* for information regarding user interface design.

2. System Overview

2.1 Background

As the administrator of the JMG Student-Site Check-in Application, most maintenance and work will be done through the interface provided by the platform upon which the app is built: Budibase. As of right now, a good amount of manual interaction is required to keep the system running for students as there are functionalities that have not been implemented yet due to the system being in its beta stages.

There are various tasks that will need to be routinely performed. All data needs to be frequently saved in another location other than the Budibase database, either locally on the administrator's machine or uploaded to a cloud file hosting service. All past data must be saved in order for JMG to either help students or JMG to use it for future endeavors. Data clean-ups must also be semi-regular, as keeping the amount of memory used within the server small will help increase system performance. All new accounts will need to be set up manually, as there isn't any functionality for automated account creation. Any missed check-ins by students, when notified by the JMG representative at the users school, will need to be manually entered into the system by the administrator, as a core functionality of the system is that the student can't check-in once the session has finished.

2.2 Hardware and Software Requirements

All that is required for the administrator to effectively maintain and run the system is to have a device that can connect to the internet (phone, computer or tablet), a web browser, and a link to access the actual application. No other software or hardware is required.

3. Administrative Procedures

3.1 Installation

Installation of the application itself is quite simple. Just visit the eventual url to the application. No other installation is required, aside from the actual hosting of the application.

3.2 An Important Distinction

There are two interfaces that will be referred to throughout the rest of this document. The Budibase development portal allows administrators and developers to edit the user base, create new applications as well as set up additional login functionality such as single sign on.

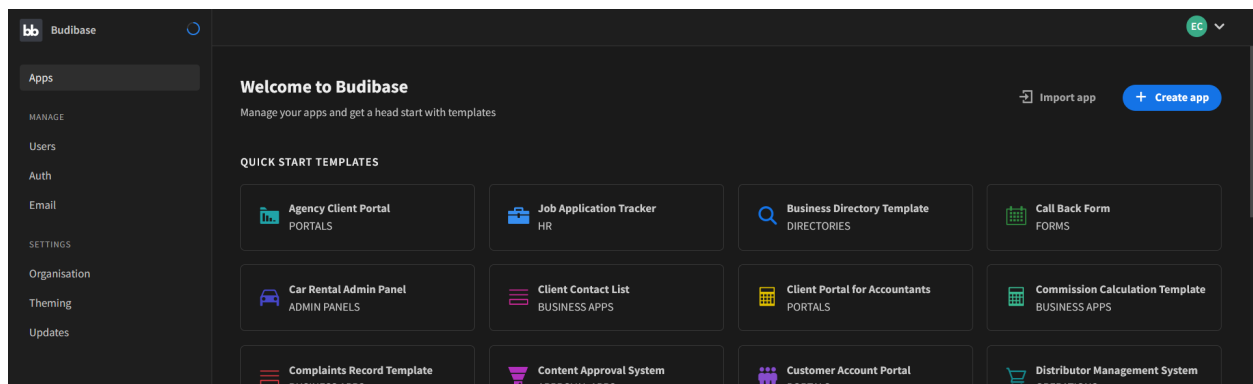


Figure 3-1: The Budibase Development Portal - This is where new applications can be created if needed, as well any additional development.

The other interface is within the application itself, this can be accessed by scrolling down to the bottom of the development portal. This is where the JMG Student Site Check-in Application and other applications (Back up copies, applications for testing, etc.).

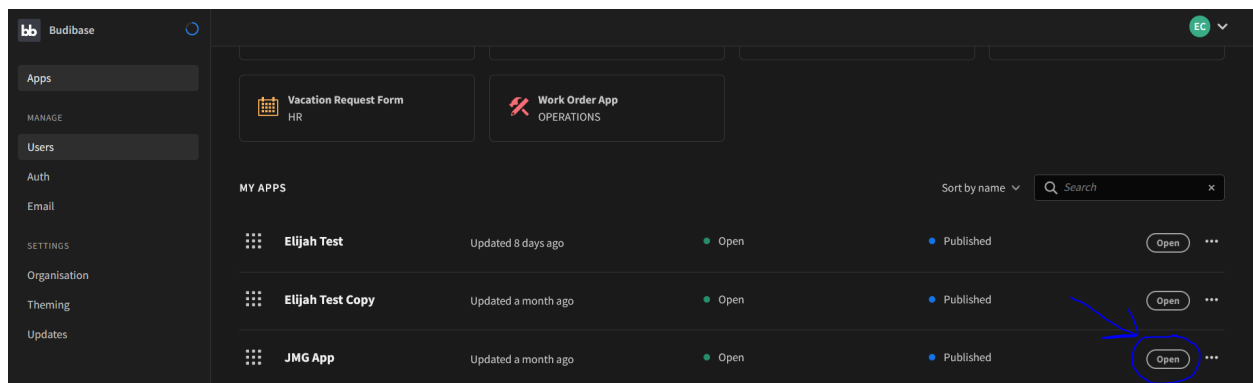


Figure 3-2: How to Access and Application - All apps created under the portal will be found at the bottom of the development portal page.

The application builder interface is used to “code” the application. Most of the administrator maintenance is done in the data tab labeled in figure 3-3. This is where all data and tables can be accessed.

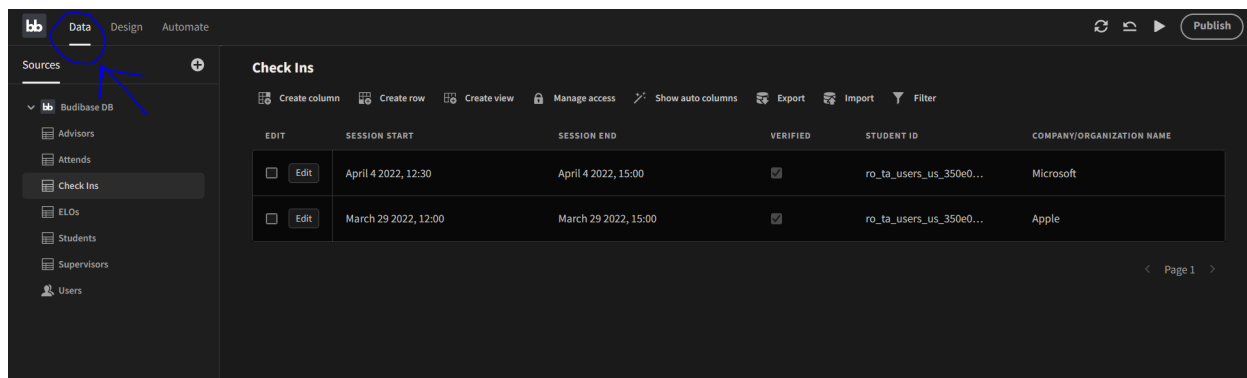


Figure 3-3: Application Interface - Most app maintenance and routine responsibilities center around the data and tables found under the data tab.

3.3 Routine Tasks

Adding Users:

To add a user to the application, the administrator must first sign in to their admin privileged account. Successful login brings administrators to the Budibase development portal. From there, click the users button on the left side navigation bar.

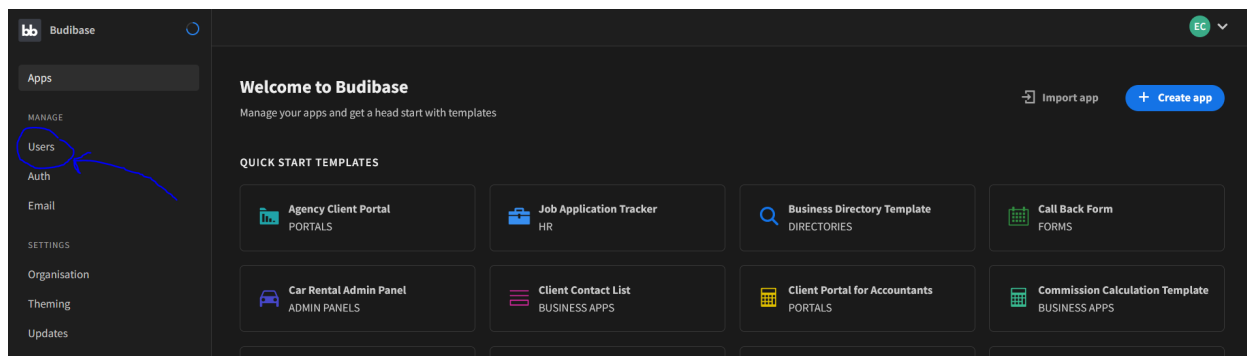
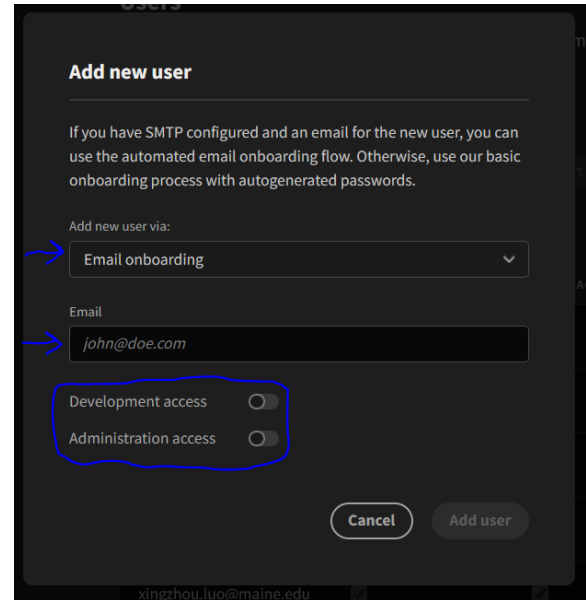
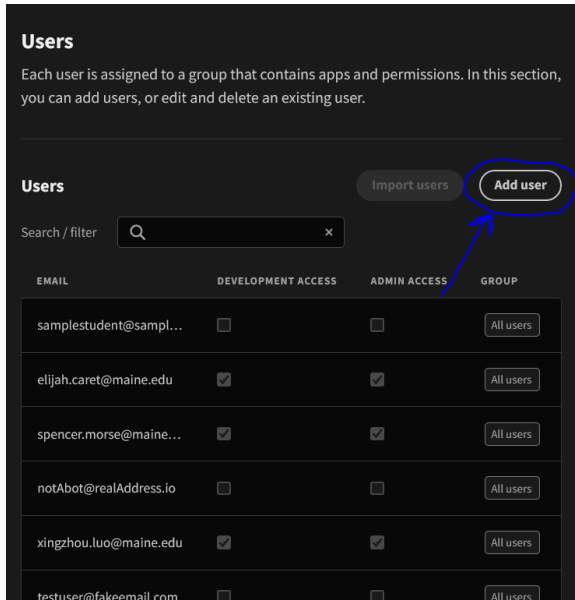


Figure 3-4: Users Tab Location - This is used to edit the user base

Click on the “Add User” button. This will bring up the window to add a new user. There are two options to add users: email onboarding or basic onboarding. Currently email onboarding is not in use so all users will be added via basic onboarding. There are two optional access permissions when creating a new user, development access and administrator access. These should only be selected if a new administrator is being added to the system.



Figures 3-5 and 3-6: Adding a User - Currently email onboarding is not functional on our application, which means the user will need to be added via basic onboarding.

The administrator will enter the email that the user will use (most likely their school email or business email). A default password will be generated for the user which they will need for their first login attempt. Once a user successfully logs in for the first time, they will be prompted to change their password to something of their choosing. Clicking the “continue” button finalizes the creation of the new user.

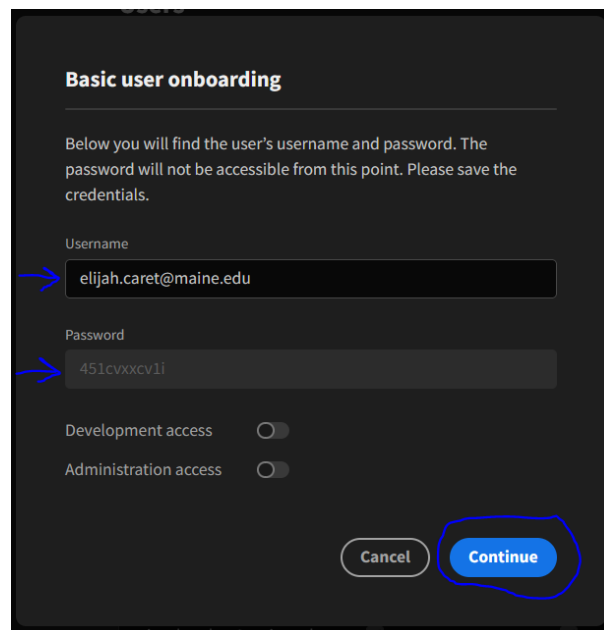
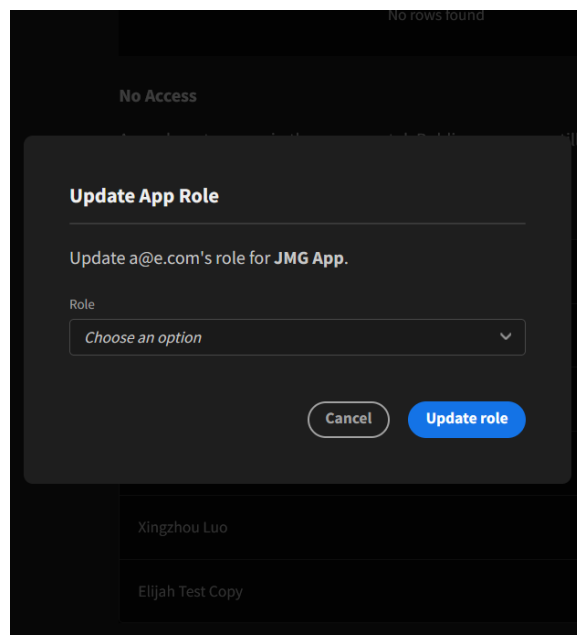
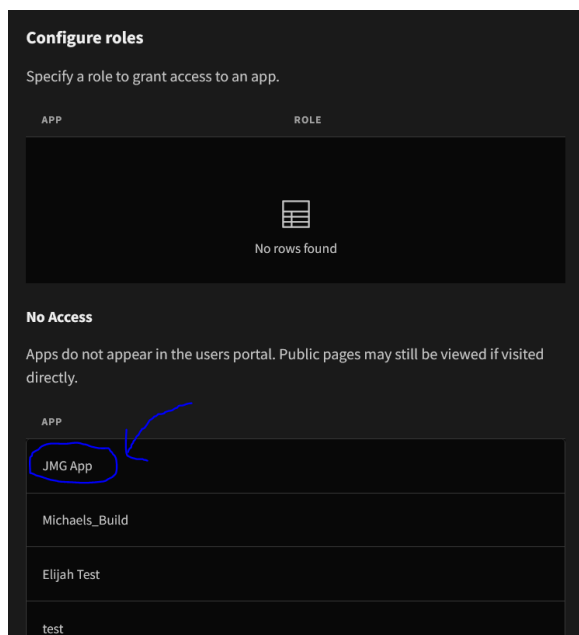


Figure 3-7: Creating a Username and Password via Basic Onboarding - The username must be an email address, which will make transition to email onboarding easier.

Lastly, the user needs to be added to the application itself. Clicking on the recently added user will open up their information. Scrolling down this page gives access to the “Configure roles” section. This section gives the administrator the ability to establish an access role for the particular application they want the user to be enrolled in. To add a user to a specific app, select the application under the “No Access” section. Then select the user’s access role for the application. In most cases, users will be given basic access to only the JMG application unless the user is an administrator. The user should now be able to access the application.



Figures 3-8 and 3-9: Configuring User Permissions for App Access - The available roles to select from include basic, power, and admin. All users will either have basic or admin access.

Removing Users:

Going back to the user information for a particular user (found under the users tab in the development portal - see figure 3-4) there is an option at the bottom of the page (scroll all the way down) to delete that user. Click the red button, and then click confirm and the user will be removed.

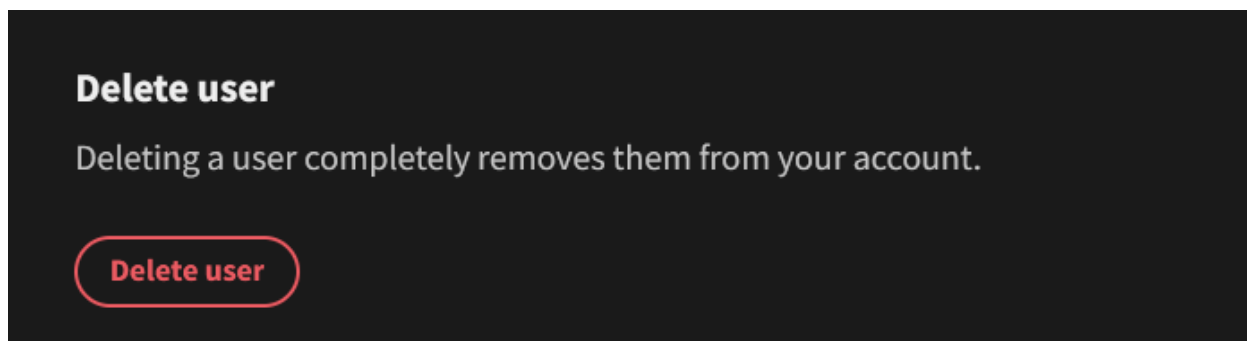


Figure 3-10: Delete User Option - Once opening up the user information, this option is located all the way at the bottom of the page (scroll all the way down).

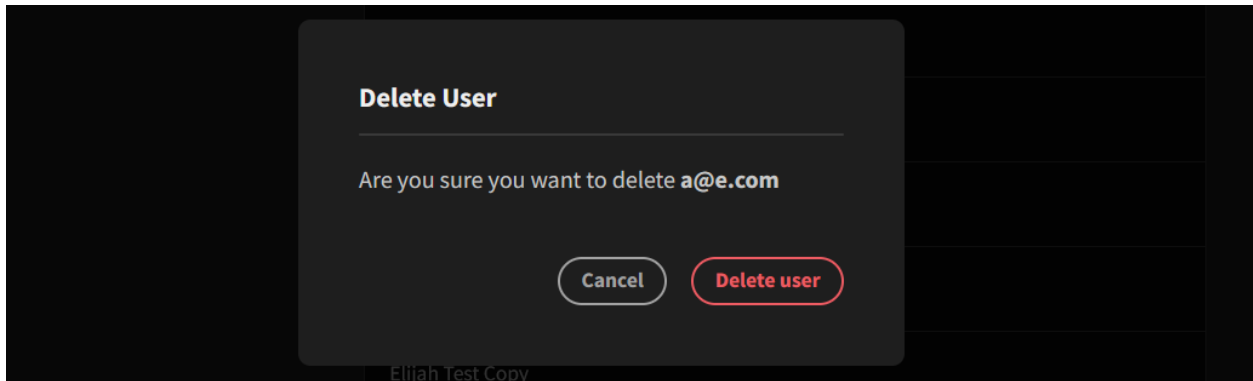


Figure 3-11: Delete User Confirmation

Fixing any Database Errors:

If any error in the database of the application is detected, such as incorrect information (according to the user whom the data is centered around), it can be fixed by manually editing the particular row that contains the data. With a fairly large dataset, the filter feature can be used to help locate the error quicker. This filter feature is located at the top of the table as shown in figure 3-12.

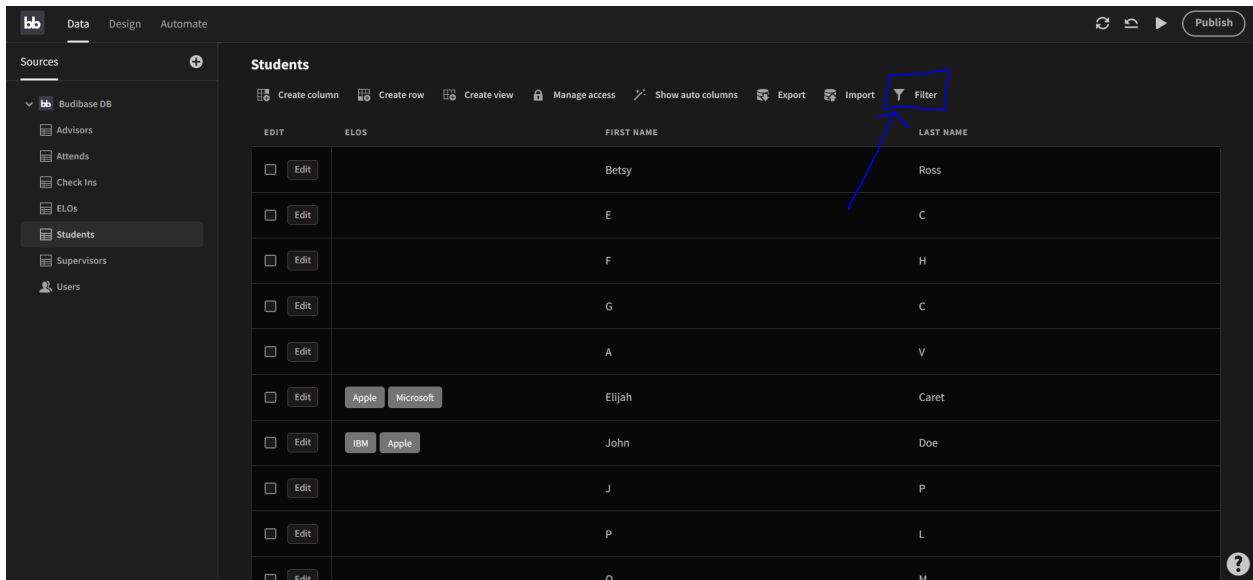


Figure 3-12: Location of Filter Feature - This feature is particularly useful to help locate a particular row.

Once the filter feature is opened, click “add filter”. Then select a particular column and value to search for. For example, figure 3-13 shows a query to search for users with the first name John.

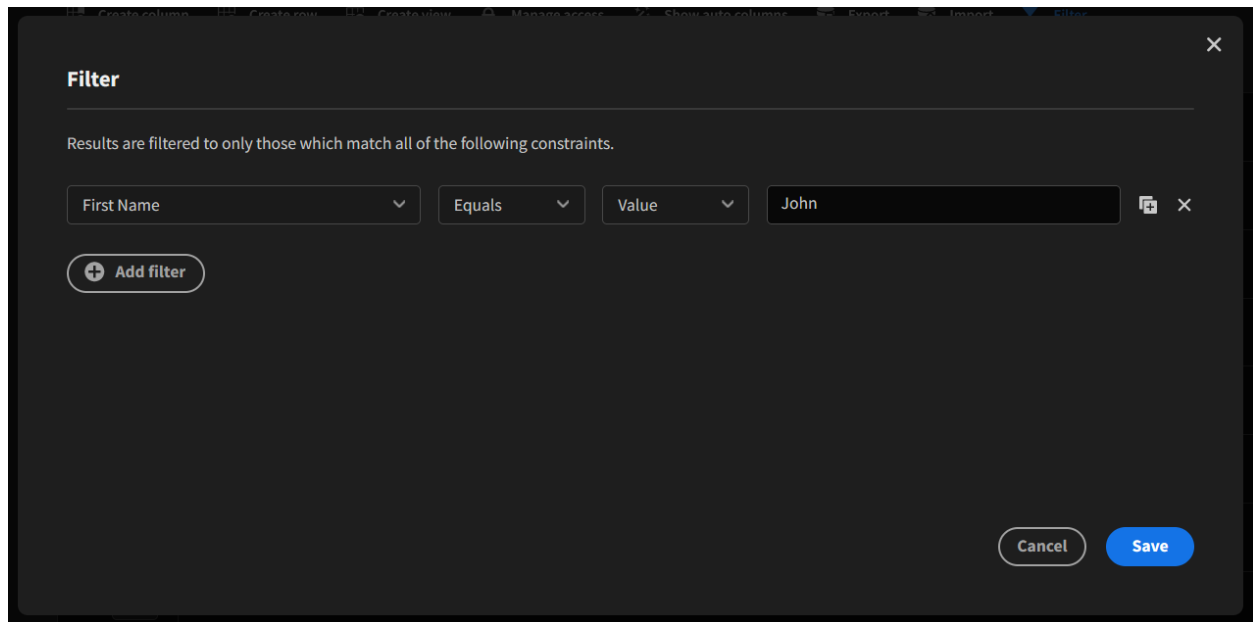


Figure 3-13: Filtering the Table - The first dropdown option is the column, while the next three options set the condition to filter the table by.

Once the proper row has been located, it can be edited by clicking on the edit button on the left side of the row. All visible attributes can now be changed based on what is incorrect. For example, in the table shown in figure 3-14 a student might have been assigned to the wrong company/organization(ELOs attribute).

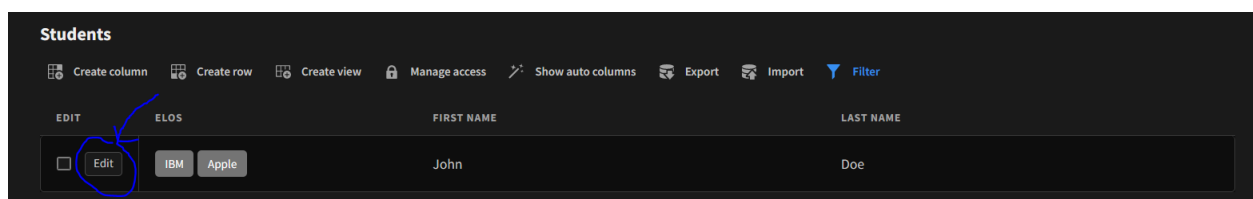
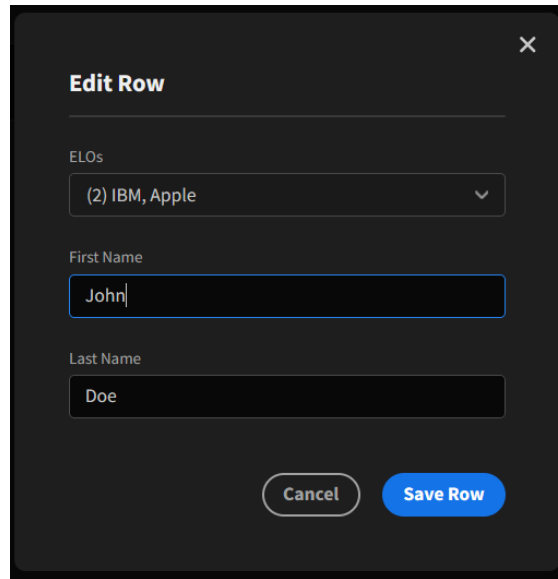


Figure 3-14: Edit Row - This option allows the administrator to edit the values of the attributes of a row.



The image shows a dark-themed 'Edit Row' pop-up window. At the top, it says 'Edit Row' with a close button (X) in the top right corner. Below this, there are three input fields: 'ELOs' with a dropdown menu showing '(2) IBM, Apple', 'First Name' with a text box containing 'John', and 'Last Name' with a text box containing 'Doe'. At the bottom, there are two buttons: 'Cancel' and 'Save Row'.

Figure 3-15: Edit Row Pop-up Screen - If John was actually attending Microsoft instead of Apple, the value of the ELOs attribute would be changed by accessing the respective dropdown bar.

Manually Adding a Missed Check-In:

If a student happens to forget to check in to their respective ELO, a JMG representative from their school, who is responsible for keeping track of their respective student's performance should contact the administrator to notify them. The administrator will then add that particular check-in, with the proper information, into the proper data table. This table is found under the name "Check-Ins" under the data tab of the application itself (Not the development portal homepage).

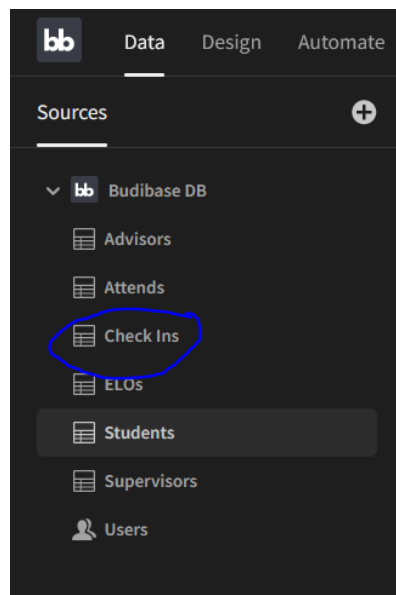


Figure 3-16: Location of Check-Ins Table - This contains all of the information about every recorded check-in on the system.

To add a row, click the “create row” icon above the table and enter the necessary information.

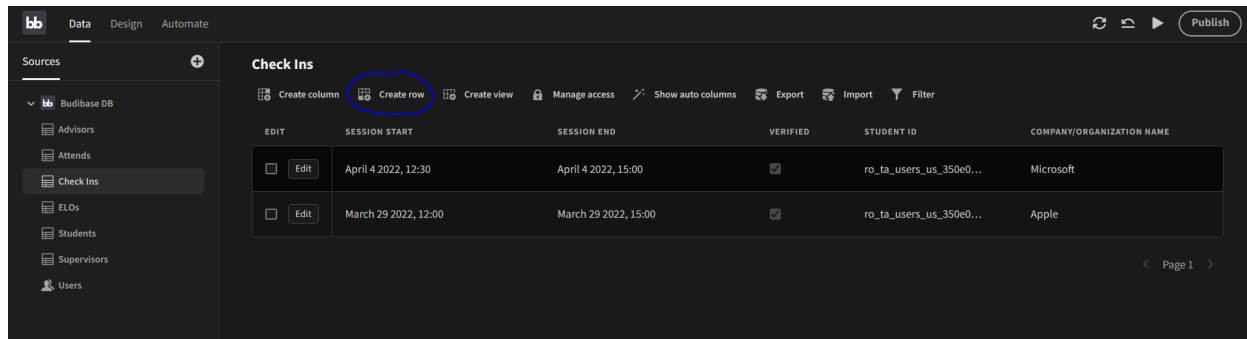


Figure 3-17: Location of Create Row Icon - This feature allows the administrator to manually add a new row to the table.

The information required here includes the time the session began, the time it ended, whether it was verified by the supervisor (this should be left unchecked, as the supervisor is responsible for verifying their respective student’s check-ins), the ID of the student and the company/organization they attended.

A screenshot of a 'Create Row' pop-up screen. The screen has a dark background and a white border. At the top, it says 'Create Row' in bold. Below this, there are five input fields: 'Session Start' with a calendar icon, 'Session End' with a calendar icon, 'Verified' with a toggle switch (currently off), 'Student ID', and 'Company/Organization Name'. At the bottom, there are two buttons: 'Cancel' and 'Create Row' (which is highlighted in blue).

Figure 3-18: Create Row Pop-up Screen - This is where all check-in information is entered (with the exception of the verified switch, which remains off).

3.4 Periodic Administration

Database Backups:

Budibase provides a feature that allows administrators to export any table to a csv/excel file. Navigate to the table you want to save. Then, at the top of the table, there will be several icons to edit the table itself. Click on the export icon as shown in figure 3-19. It can either be exported as a csv file or json file if needed. After that, it is recommended that this file gets backed up to either a cloud storage service or on a flash drive. This is usually done at the end of the student's semester/term.

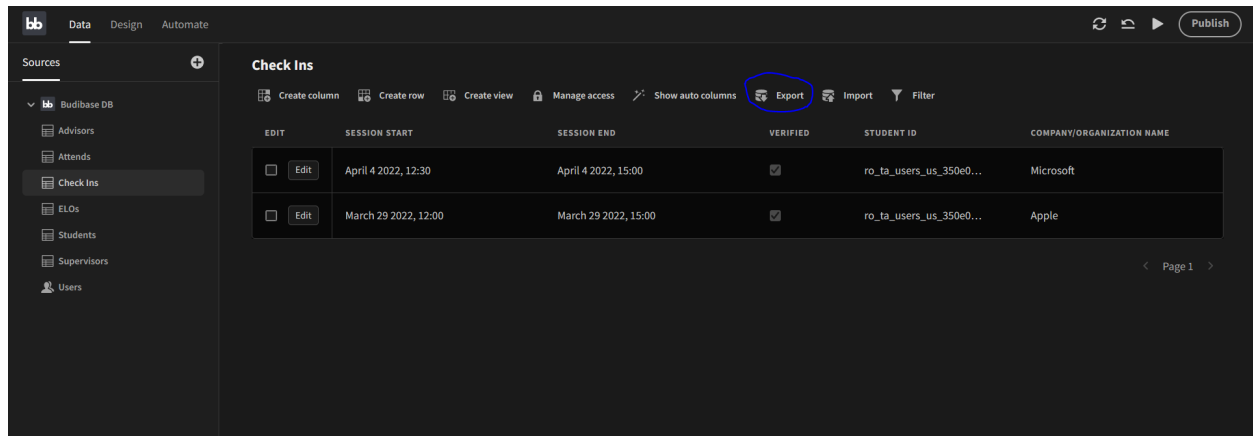


Figure 3-19: Location of Export Icon - This feature is used to download any table in the application database to a local machine.

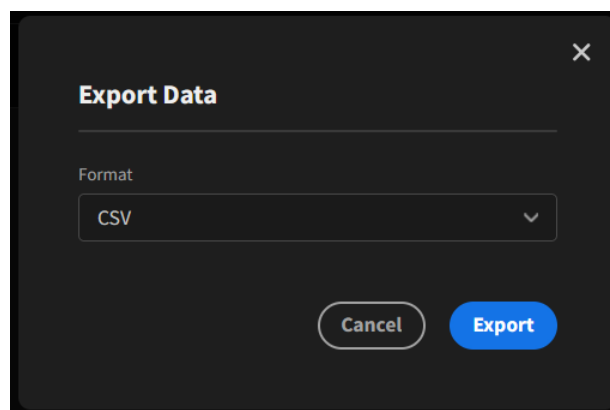


Figure 3-20: Choose Format to Export Table - The dropdown allows for the selection of either a csv or json file.

Database Cleanup:

The administrator must ensure that all desired data has been backed up and saved before this is done. The only way to clean up the database is to remove each row individually. The Cyber Cookie team is currently investigating a potential alternative to this and will update this document accordingly. As of now however, the cleanup is done through deleting rows. To delete the row, select it by clicking anywhere on it, or select the checkbox at the left side of the row.

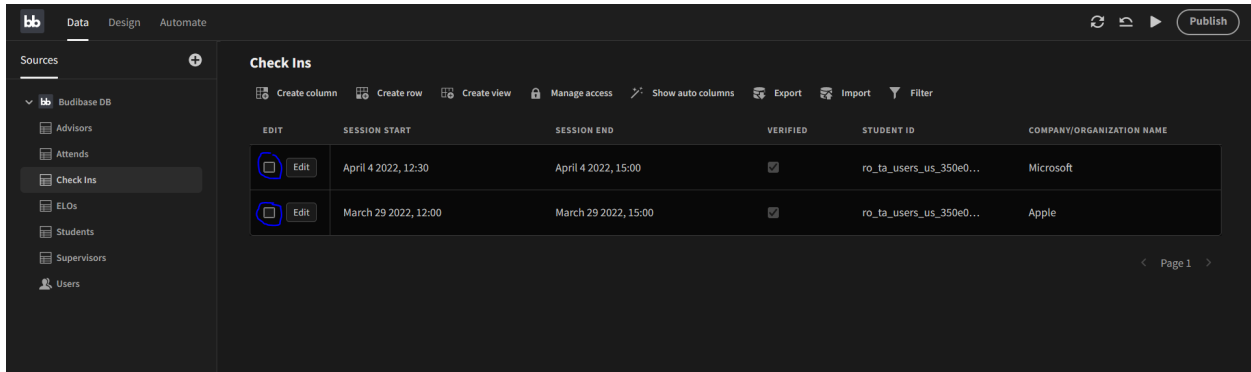


Figure 3-21: Selecting Rows - It can be done either by selecting the checkbox to the left of each row or by clicking anywhere on the row itself.

A new option will appear with the other icons at the top of the table labeled “Delete (x) row(s)”. Click on this option to remove the selected rows.

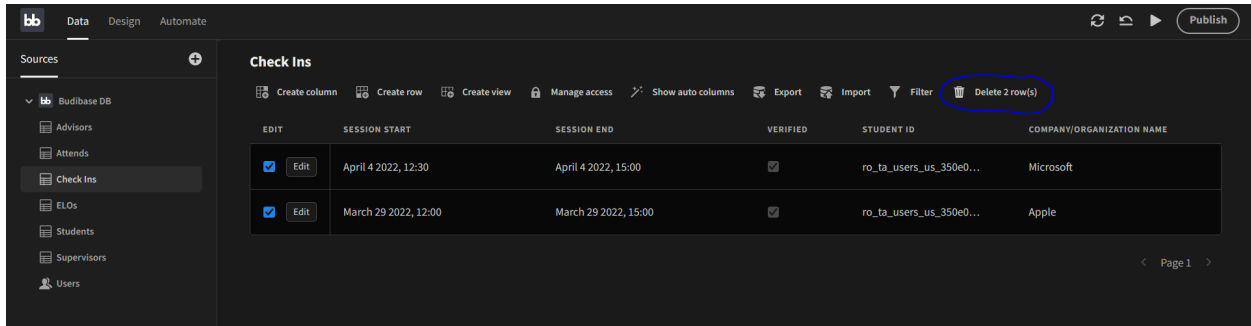


Figure 3-22: Deleting Rows - Once the desired rows to deleted have been selected, a new icon will appear at the top of the table as circled

3.5 User Support

Currently any user support can be directed to any members of the Cyber Cookie team.

Elijah Caret: elijah.caret@maine.edu or epcaret@yahoo.com

Spencer Morse: spencer.morse@maine.edu

Michael Ferris: michael.ferris@maine.edu

Xingzhou Luo: xingzhou.luo@maine.edu

Eventually support will be transferred over to the JMG ELO technologist via a support email (elohelp@jmg.org for example) as well as a phone number.

04. Troubleshooting

4.1 Dealing with Error Messages and Failures

Build Breaks:

In the event of a broken update to the system being published. Budibase provides a version control system that will allow administrators to revert back to a previous version of the application. At the top right of the screen, there are three icons as well as the publish button. Selecting the middle icon will revert the application back to the previously published version.

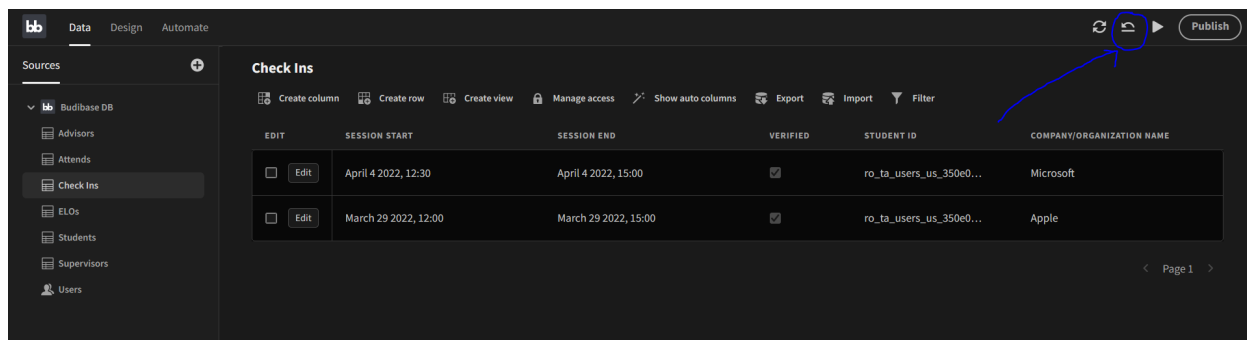


Figure 4-1: Version Control Feature - Most system breaking errors are mitigated by clicking this icon, which reverts back to the previous working version

Other helpful tips here are to run/make any new updates on an unpublished private copy of the application. It's also important to save all data prior to publishing a new update in the event of a system failure.

4.2 Known Bugs and Limitations

The user interface of the system is very limited and basic to serve functionally.

There is currently no student performance assessment functionality implemented, although that will change before acceptance testing.

Table linking is not fully fleshed out, as the Cyber Cookie team is still trying to understand how the database works in that respect. Relationships are the definite feature to use to link tables, and more specifically columns together, but there has been a challenge trying to utilize these relationships in the design module of the application.

The established roles have also not been fully implemented, the student and supervisor roles are in place, but school supervisor/JMG representative roles have not. This is also something that will be most likely finished by acceptance testing.

Functionality to adjust for phone screens is not finished, but is currently under development. This simply was something that appeared later on the team's checklist for things to complete.

Appendix G: User Guide

JMG Student Site

Check-in Application

User Guide

Client



JMG

Lanet Anthony, Samantha Brink

Developer



Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine

April 20, 2022

Version 1.0

TABLE OF CONTENTS

INTRODUCTION	90-92
Intended Readership	90
Applicability Statement	90
Purpose	90
How to Use this Document	91
Related Documents	91
Conventions	91
Problem Reporting Instructions	92
OVERVIEW	93
INSTRUCTIONS	94-100
REFERENCE SECTION	101
ERROR MESSAGES AND RECOVERY PROCEDURES	102

INTRODUCTION

The JMG Student Site Check-In Application is a web service that aims to automate the process of JMG students notifying teachers of their attendance at events outside of school counted towards course credit. This is a capstone project for the Cyber Cookie team, which includes Elijah Caret, Michael Ferris, Xingzhou Luo, and Spencer Morse, in partial fulfillment of the Computer Science BS degree for the University of Maine. The deliverable of this project practices the development cycle of industrial standards. During the previous semester, the Cyber Cookie team had established system requirements, architecture definitions, and user interface designs. The goal for the team this semester is to implement and integrate these modules into the desired functional system with supportive documentation.

INTENDED READERSHIP

There are 4 kinds of users who will be involved in using the system. Included are 3 end users and 1 system operator. The 3 end users include student users, which are the main users of the system and will use the check-in functionality of the system. The second end user is the supervisor user, these are the users who act as supervisor or company employee supervising the student at their session. They mainly will use the system to provide feedback to students on performance as well as verify their check-ins. Advisor users are the 3rd type of end user, who serve more as monitors for student performance. All end users are not required to have any experience in web app development to use the application and can find all information related to their use in the instruction section of this guide. Administrators serve as the system operators who are responsible for maintaining data, handling any issues, registering new users, and maintaining data and system back ups to name a few. Administrators also may or may not possess experience in web app development and maintenance. All responsibilities are laid out in more detail in the referenced Administrator Manual.

APPLICABILITY STATEMENT

This issue of the User Guide (Issue 1), applies to version 1.0 of the application software.

PURPOSE

The purpose of this application is to allow students attending ELO's to have a digital place where they can check in to those locations. This application also allows for site supervisors to verify and track student attendance, as well as supply feedback on said students.

The purpose of this User Guide is to assist users of the student check-in application in any possible confusion or errors they may encounter.

HOW TO USE THIS DOCUMENT

Important sections of this document include:

Overview: A general overview of the system and its capabilities. The intent is for users reading this document to get a good grasp of what they will be able to do within the application in general, the instructions go into more detail regarding what they can do.

Instructions: This section will provide you with written instructions and screenshots on important application features. It is intended to be used by students or site supervisors looking for guidance on application features. The reference section will specify how users can input data.

Reference Section: This describes the data involved in more detail and the user's influence on how that data is manipulated or sent. It gives more context to the instructions provided in the instructions section.

Error Messages and Recovery Procedures: This section covers any potential error messages that a user may receive and how to deal with them. The instructions in the instruction section briefly mention the important ones.

RELATED DOCUMENTS

Num	Title	Author	Date	Issue
1	JMG Student Site Check-In Application Administrator Manual	Elijah Caret, Michael Ferris, Xingzhou Luo and, Spencer Mores	04/06/2022	V1.0

CONVENTIONS

In this document you will often see naming conventions for buttons on the app that look like this: “<name of button>”. This refers to a clickable button on the app and the word or phrase inside of the quotations is what the button is labeled on the app. For example, if the instructions say to click the “Submit” button, it is referring to a button like this on the app.

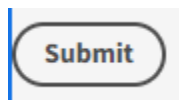


Fig. 1 - Submit Check-In Button

If something is in italics, it often means that it is a note or potential warning regarding a defined portion of the document or application.

PROBLEM REPORTING INSTRUCTIONS

Users that run into any issues that have not been listed in this document should email either their site supervisor or system administrator and inform them of the problem.

In future versions of the application there could potentially be a Help button within the application where users can leave a help request that can be sent to either the site supervisor or administrator.

OVERVIEW

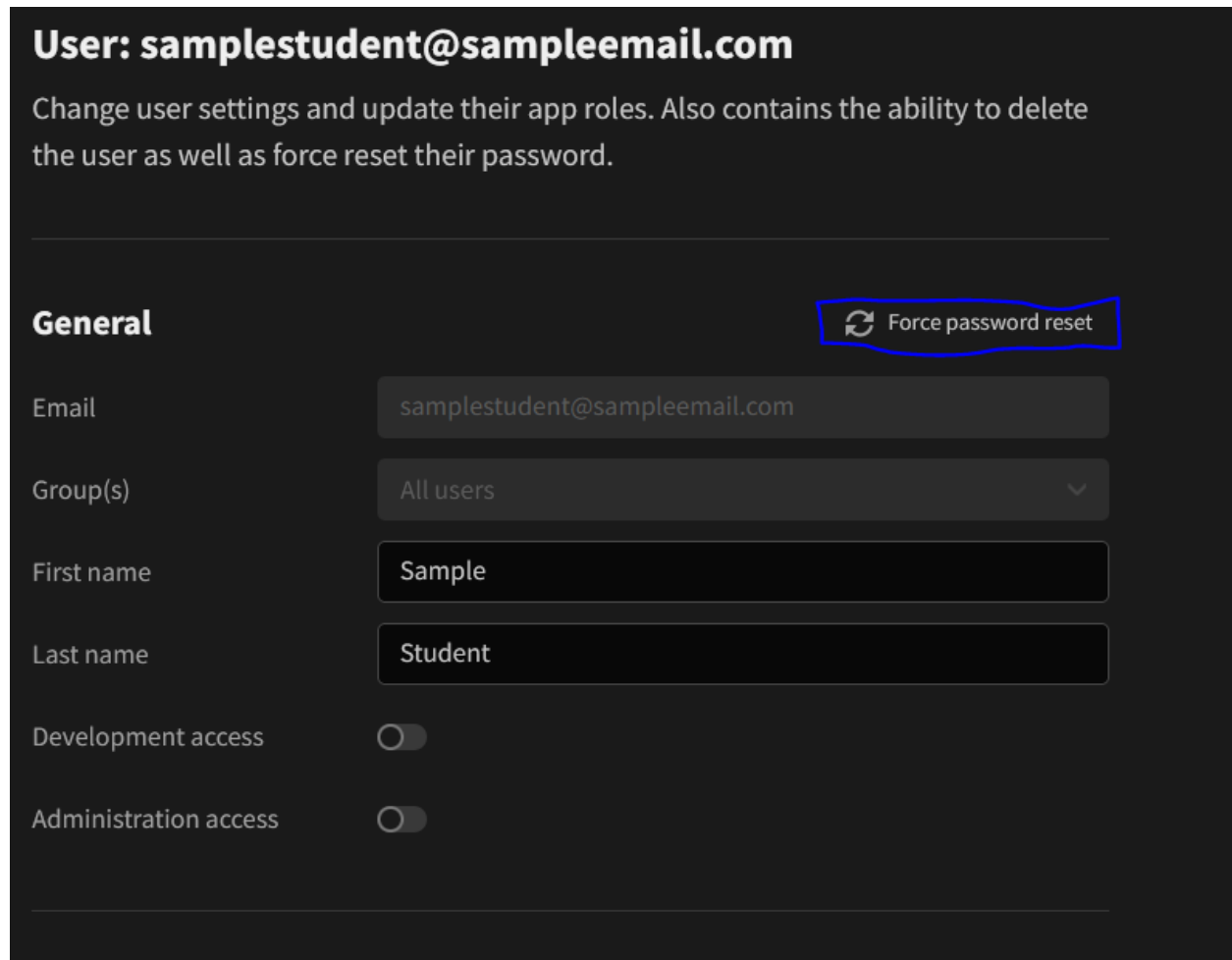
The JMG Student Site Check-In Application allows users, in particular students, to electronically record any extended learning opportunity that they attend. This is done through simply clicking on the “Check In” button located on the homepage, and specifying a few details in a form such as session dates via a calendar picker and the organization involved via a drop down bar. JMG representatives at the student’s school will be able to look at their student’s attendance and performance by clicking on said student on the homepage. Supervisors at the various companies and organizations students attend for sessions will be able to view their student’s check-ins by clicking the “View Check-Ins” button. They will also be able to verify that the student was present by simply clicking the “Verify” button under each check-in. Each check-in will also have a performance analysis assessment to fill, the form is opened by selecting the “Submit Performance Analysis” button, located under each check-in.

INSTRUCTIONS

Administrator Responsibilities:

All application administrators should refer to the JMG Student Site Check-In Application Administrator Manual for instructions on system maintenance.

One item that was not mentioned was account recovery. This can be done via the force password reset option. This is located by clicking on the users tab in the development portal, and then clicking on the user whose password needs to be reset. In the top right corner of the general section, there is an option to force a password reset.



User: samplestudent@sampleemail.com

Change user settings and update their app roles. Also contains the ability to delete the user as well as force reset their password.

General ↻ Force password reset

Email

Group(s)

First name

Last name

Development access ☐

Administration access ☐

Figure 1: Location of Password Reset Option - Only use this when the user has exhausted all other options.

A new password will be randomly generated, which will be saved and given to the user.

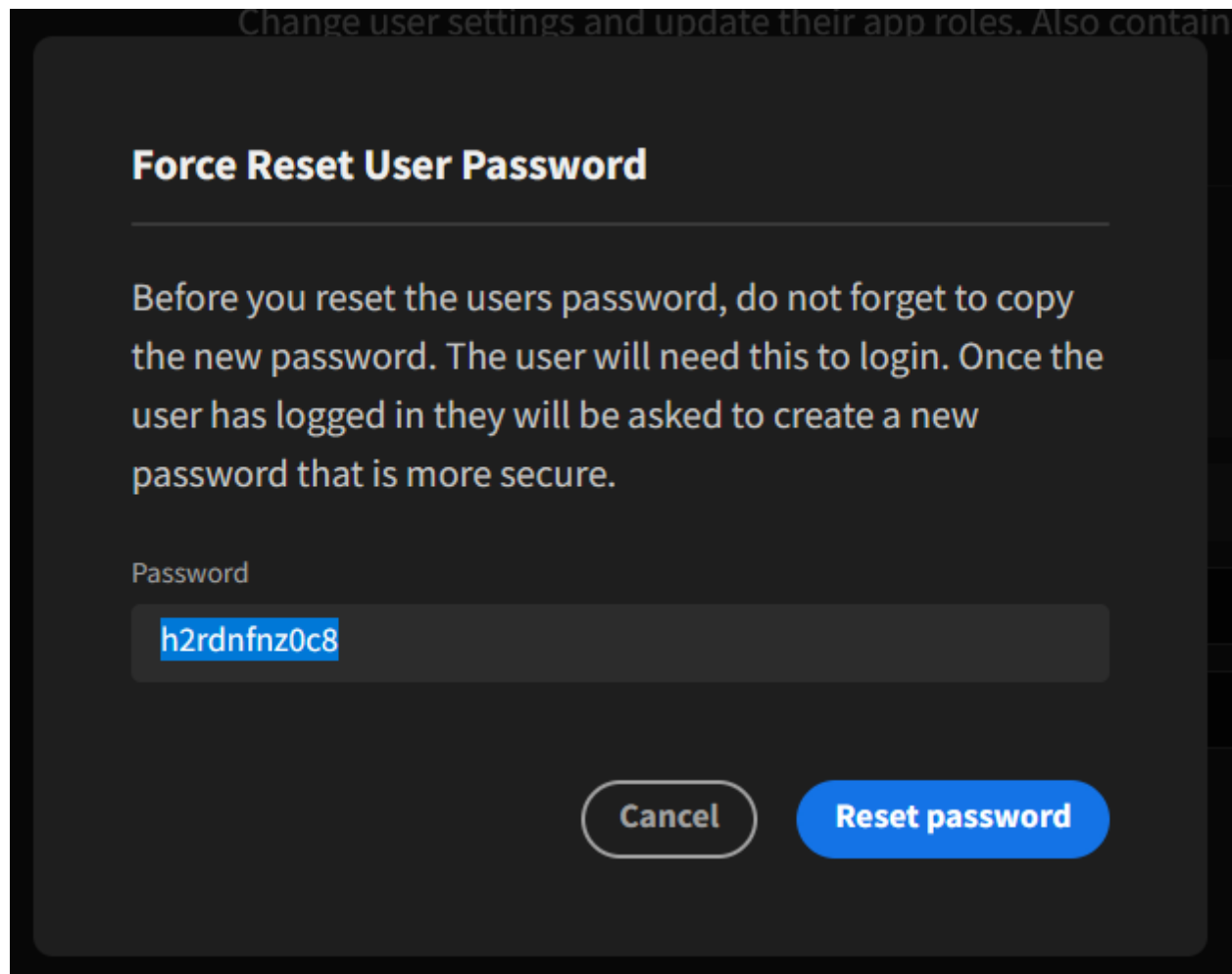


Figure 2: New Random Password - Save this password and then click “Reset password”


The user can then login with this password, and then be prompted to change to a new password of their choosing.

Login/Logout:

Users will be required to authenticate themselves via a login page. Login is done via a standard authentication of an email address and password. This section will go over two forms of login. One for first time users and one for regular users.

First Time Users and Password Resetting:

A system administrator will set up a new user's account with a given email address and both the new users and the users that are resetting their password are given randomly generated password. In both cases, they must include these credentials in the appropriate text field locations to log in.



Sign in to JMG Student Site Check-In

Sign in with email

Email


Password

[Sign in to JMG Student Site Check-In](#)

[Forgot password?](#)

Figure 3: Entering Credentials - New users will have a random password given to them by the system administrator

Once the user is logged in, they will be prompted to create a new password of their choosing. Once again they should enter the new password in the given text field. They will also be asked to re-enter the new password.



Reset your password

Please enter the new password you'd like to use.

Password

Repeat Password

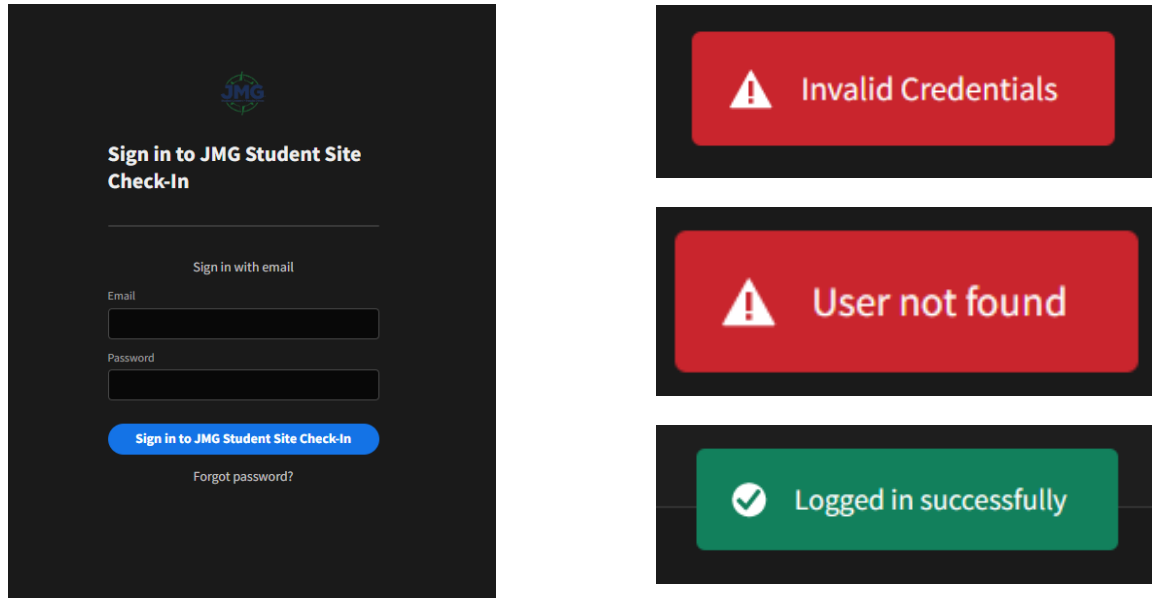
[Reset your password](#)

Figure 4: Creating/Resetting Password - This screen shows up in both cases.

Regular Users:

Upon arriving at the login page, regular users will enter their email address and password that they had created upon logging into the system for the first time.

Indication of login status will be displayed as a text bubble at the top of the screen, indicating either a successful login (green) or a failed login (red).



Figures 5 - 8: Login Screen and Login Status - Three different messages regarding the login attempt are likely to be seen. The two error (red) messages will be explained in more detail in the errors and recovery procedures section

Password Reset:

As of version 1.0, password resetting is not automated and must be initiated by a system administrator. Users should contact their system administrator to ask for a forced password reset.

Logout:

To log out, users should click the icon in the top right corner, opening a dropdown menu. Then users can click the “Logout” option to logout.

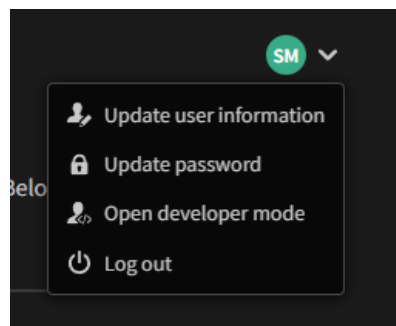


Figure 9: Logout - Standard users will not have access to the open developer mode option.

Check-In:

Upon arriving at the homepage, student users will most likely be adding a new check-in to the system. To do this, they should click the “Check-In” button under the list of their enrolled organizations. A pop up screen will appear that contains a small form for the student to fill out. This form includes the start and end time of the session. These can be edited by clicking on the field, which opens a calendar picker. The other field item is the organization they are checking into, which is a dropdown selection.

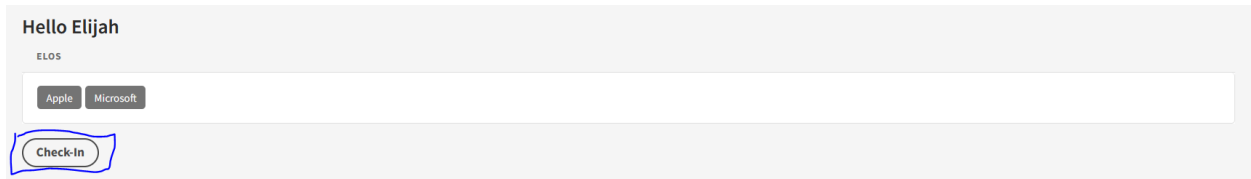


Figure 10: Check-In Button Location - Only student users will have access to this button

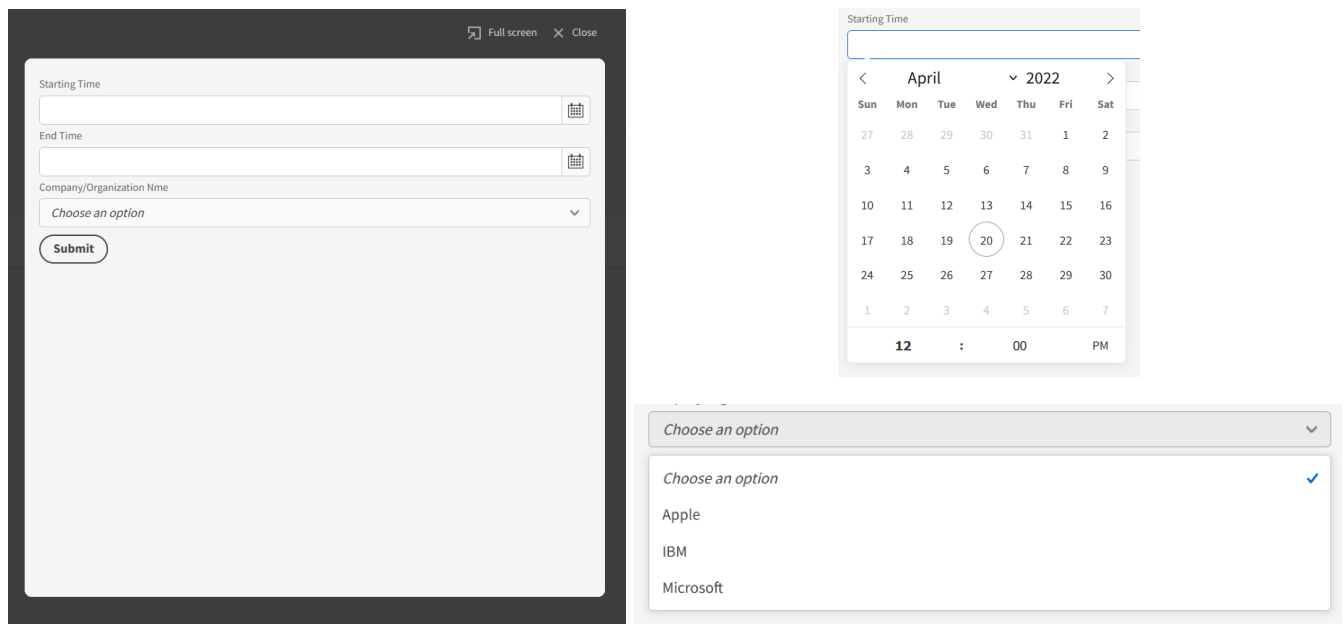


Figure 11 - 13: Check-In Form, Calendar Picker and Drop Down Option - Students will use these futures to submit a check-in

Once all information has been entered, users must click the “Submit” button. An additional pop up screen will appear, asking the user if they are sure they entered the correct information. If they have verified it to be correct, they can click the “Confirm” button. A text bubble will appear at the top of the screen to indicate the status of the information being sent. It will either say that the row (which the data from the form is saved in) was successfully saved (green), or failed to save (red).

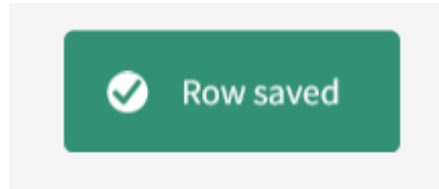
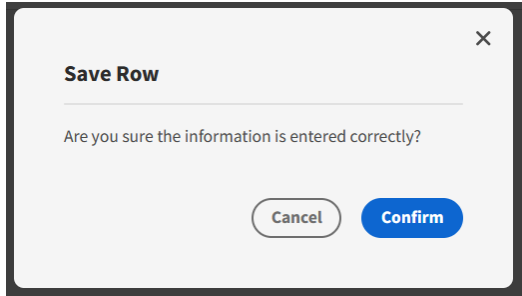


Figure 14 and 15: Confirmation and Row Saved Indicator - Users should make sure that they entered the correct information.

Viewing Check-In History:

All users will be able to view check-in history. What gets displayed depends on the type of user. JMG advisors/teachers will only see check-ins for students attending their school, supervisors will be able to see only check-ins for the students that they supervise/are enrolled in that particular ELO. Students will only be able to see their own check-ins. All users can access this through clicking the “View Check-Ins” button on the homepage.

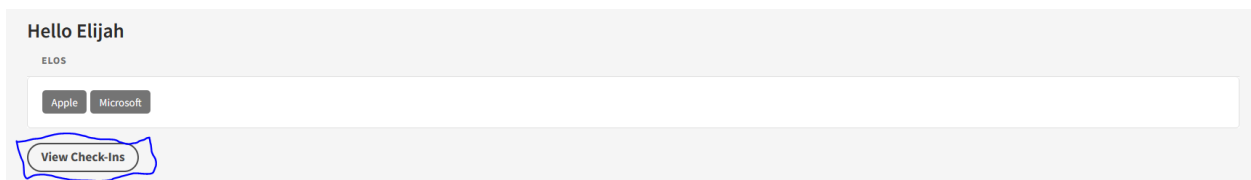


Figure 16: View Check-Ins Button Location - This screenshot is from the perspective of a supervisor.

All users may also see feedback on each check-in depending on whether or not it has been submitted by the supervisor. Students will see their own feedback, while supervisors and advisors will be able to see their respective student’s feedback. Students will not be able to view other students’ feedback.

Verify Check-Ins:

Supervisors will have the ability to verify check-ins once they have entered the page showing all check-ins. Under each check-in description, there will be a button to verify that particular check-in labeled “Verify”. Clicking this button will open a pop up screen, asking the user if they are verifying the correct check-in. Upon clicking the “Verify” button. A speech bubble will appear at the top of the screen to indicate the status of the information being sent (refer to figure 14). It will either say that the table (which the data from the form is saved in) was successfully saved (green), or failed to save (red).

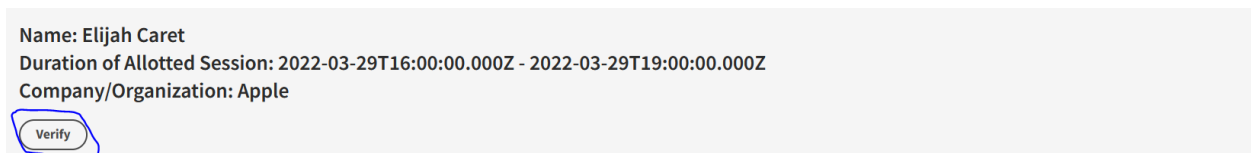


Figure 17: Verify Button Location - This button will be listed under each check-in.

What actually happens here is that an attribute associated with that particular check-in labeled “Verified” will be set to true once the data is saved. When a check-in is already verified, the “Verify” button will be disabled.



Figure 18: Verify Button Disabled - This will disable itself upon clicking it and confirming the verification. Refreshing the page isn't required to see the change.

Submit Performance Assessment:

Warning/Note: Version 1.0 does not have this feature, this is a layout of how it is going to work. It will be implemented in the next version.

The supervisor will be able to give the student feedback on their performance. To do this, they will open the check-in history page and select the “Submit Feedback” button located under each check-in description. A pop up screen will appear, allowing the supervisor to submit their feedback via a textfield. Once they click submit, a verification screen will appear, asking them if they are sure they want to submit. Clicking “yes” will lead to a text bubble at the top of the screen indicating the status of the submission where it will either be successful (green) or have failed (red).

REFERENCE SECTION

Send Check-In:

All check-in entries have 5 attributes: start time, end time, company ID, student ID, and whether or not the check-in was verified. The start time and end time are represented as dates, which consist of DD/MM/YYYY and the time (military). Company ID and student ID are represented as 7 digit numbers. The verified attribute is simply a true or false value. Start time, end time, and company name are all required to be entered by the user, the rest is filled out by the system based on the information given. Each check-in, once submitted, is sent to the check-ins table of the database.

All entries will be valid and sent to the database. This is a potential issue as any date can be put into the system and all information can be left blank. Any organization can also be selected from the drop down menu. A warning for administrators is that some information sent to the database will not be consistently accurate, and the accuracy of data is based more on the merit of the users.

Verify Check-In:

After the button click, the respective row in the table is updated. In particular, the “verified” attribute is updated to “true” in the table. No particular format is required from the user other than the button click.

Submit Performance Assessment:

Note: This feature has not been implemented in version 1.0, but will be implemented in future versions.

The data obtained from the performance assessment field is text and is stored alongside each check-in when completed as an additional attribute.

ERROR MESSAGES AND RECOVERY PROCEDURES

“User not found” (Refer to Figures 5 - 8) - This means the email you entered is incorrect. Users encountering this error should make sure they are using the correct email address. Frequent mistakes include mistyping the address itself as well as adding any additional spaces beyond the name. Users who have tried these fixes and are still not successfully being found should contact the system administrator.

“Invalid Credentials” (Refer to Figures 5 - 8) - This means that the password entered was incorrect. Users should check to make sure they entered the password correctly. A common mistake is to add an additional space at the end that wasn’t meant to be there. If the issue persists, users should contact the system administrator for a password reset.

“Row Failed to Save” - This will appear either when the user attempts to submit a check in or a supervisor attempts to verify a check-in. Most of the time, this occurs with internet connection problems. Sometimes a user may be logged out due to the session period expiring, which will cause data to not be updated. Users should contact the system administrator if the above causes aren’t present.

GLOSSARY

Advisor - User that represents the JMG representative at the student’s institution.

Attribute - A subcategory of a data table. For example, the check-in data table has a “verified” attribute that indicates whether or not the check-in has been verified.

ELO - Extended Learning Opportunities - An opportunity for students to gain course credit while working at an organization or company whether that be via an internship or job shadow.

Field - A source of user input data. For example, a dropdown bar is a type of field.

Student - Type of user, represents the student that will be enrolled in various ELOs and doing the check-ins

Supervisor - User that is responsible for mentoring the student at the ELO. Usually an employee/member of the company/organization.

Appendix H: Project Poster

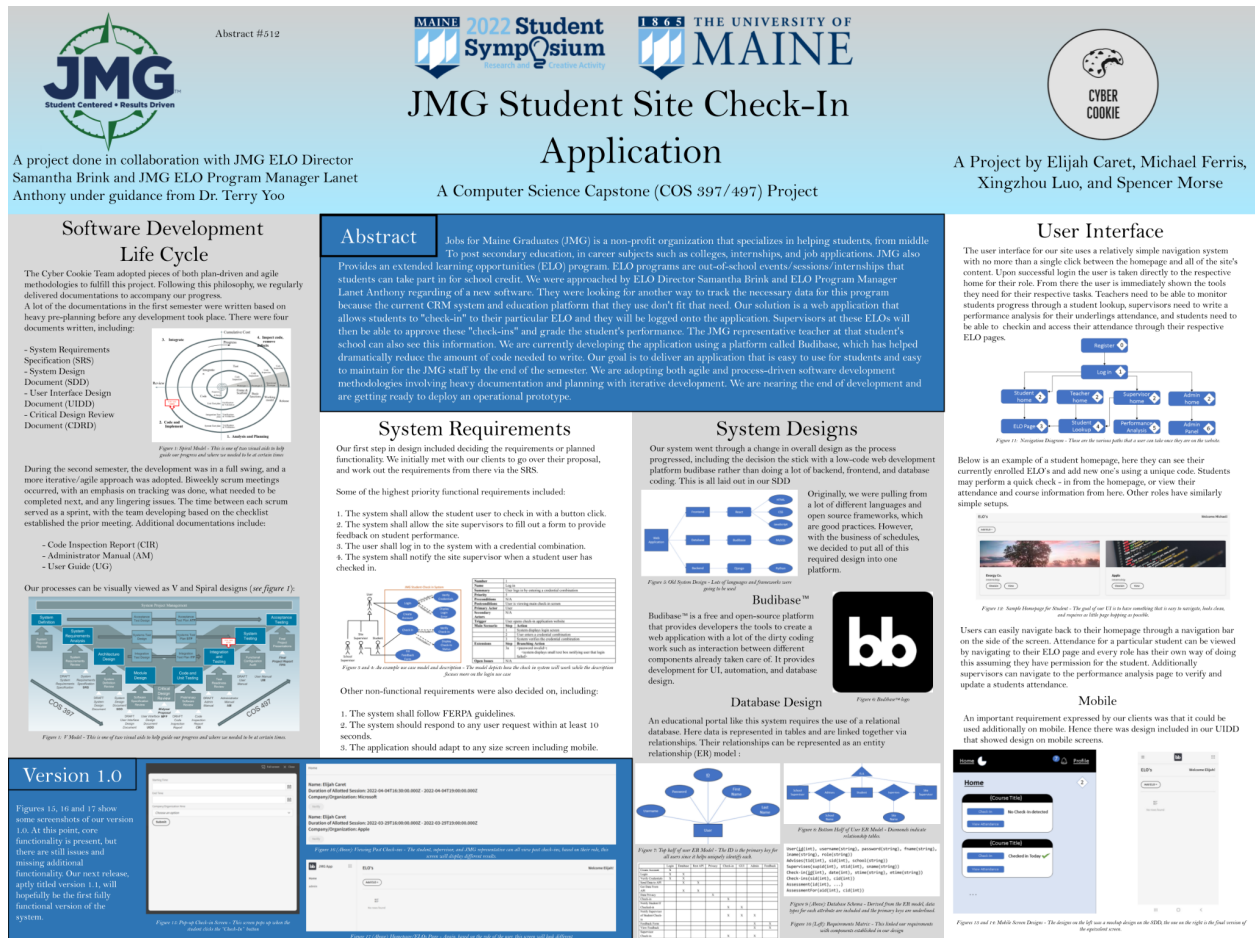


Figure H-1: UMaine Student Symposium Project Poster - The team presented their project at the University of Maine Student Symposium with this 36 x 48 in poster.

Appendix I: Team Review Sign Off

By signing below, all members agree that they have reviewed this final report and agree on its content and format.

Name: Elijah Caret	Signature: <u></u>	Date: 5/06/2022
Name: Michael Ferris	Signature: <u></u>	Date: 5/06/2022
Name: Xingzhou Luo	Signature: <u></u>	Date: 5/06/2022
Name: Spencer Morse	Signature: <u></u>	Date: 5/06/2022