

JMG Student Site Check-in Application

Critical Design Review Document

Clients

Lanet Anthony, Samantha Brink, JMG

Developer



Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
December 13, 2021
Version 1.1



JMG Student Site Check-In Application User Interface Design Document

Table of Contents

| | | |
|------|--|----|
| I. | Introduction | 3 |
| | 1. Purpose of the Document | 3 |
| | 2. References | 3 |
| II. | JMG and the Desire for a Student Check-in System | 4 |
| | 1. About JMG | 4 |
| | 2. The Problem | 4 |
| | 3. Overview of the Goal Product | 4 |
| III. | JMG Student Check-in Application Design and Planning | 5 |
| | 1. Requirements | 5 |
| | 2. System Design | 6 |
| | 3. User Interface | 11 |
| IV. | Future Plans, Additional Information, and Concluding Remarks | 13 |
| | 1. Security and Privacy Concerns | 13 |
| | 2. Future Plans | 13 |
| | 3. Timeline of Deliverables | 14 |
| | 4. Team Roles | 14 |
| | 5. Concluding Remarks - Supporting JMG Students | 15 |
| | Appendix A: Document Contributions | 16 |
| | Appendix B: System Requirements Specification | 17 |
| | Appendix C: System Design Document | 32 |
| | Appendix D: User Interface Design Document | 46 |

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

See “*System Requirements Specification*” (Appendix B), “*System Design Document*” (Appendix C), and “*User Interface Design Document*” (Appendix D) for further information.

Salesforce. *Rest API Developer Guide*. Retrieved December 13, 2021.
https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/intro_rest.htm

Budibase. *Budibase Docs*. Retrieved December 13, 2021.
<https://docs.budibase.com/>

JMG. *What is JMG?*. Retrieved December 13, 2021.
<https://www.jmg.org/about-jmg/what-jmg>

JMG. *Cast the Vision: Designing Maine’s Model*. Retrieved December 13, 2021.
<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.

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 these opportunities to apply for other jobs and colleges further down the road.

3. Overview of the 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. 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.

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 an assessment button for Supervisors to access their underlings.
- The application must provide a means for roles to search for users and courses under their jurisdiction.
- Course creation must be communicated to salesforce.
- 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 rate 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, or admin which will control what data you're 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 will 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.

Prior to that however, additional testing will be done beforehand to ensure that the application has met all of the requirements in terms of functionality and performance. For example, in order to test to see if the login requirement is met, a fake user will be created. They will first register a new account and observe both the response time of the system, as well as whether or not the system responded correctly to the user's input. We intend to use this mock/stub account to examine each functional requirement.

For non-functional requirements. Things such as stress testing and timing will be needed as criteria for determining if the requirement has been met. For others such as the second requirement listed in section 3-1-b, the application may have to be run on a mock-up mobile screen (whether through software or an actual device) and a computer screen.

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.

a. High-Level Architecture

At the highest level, the system will consist of 4 major parts interacting with each other.

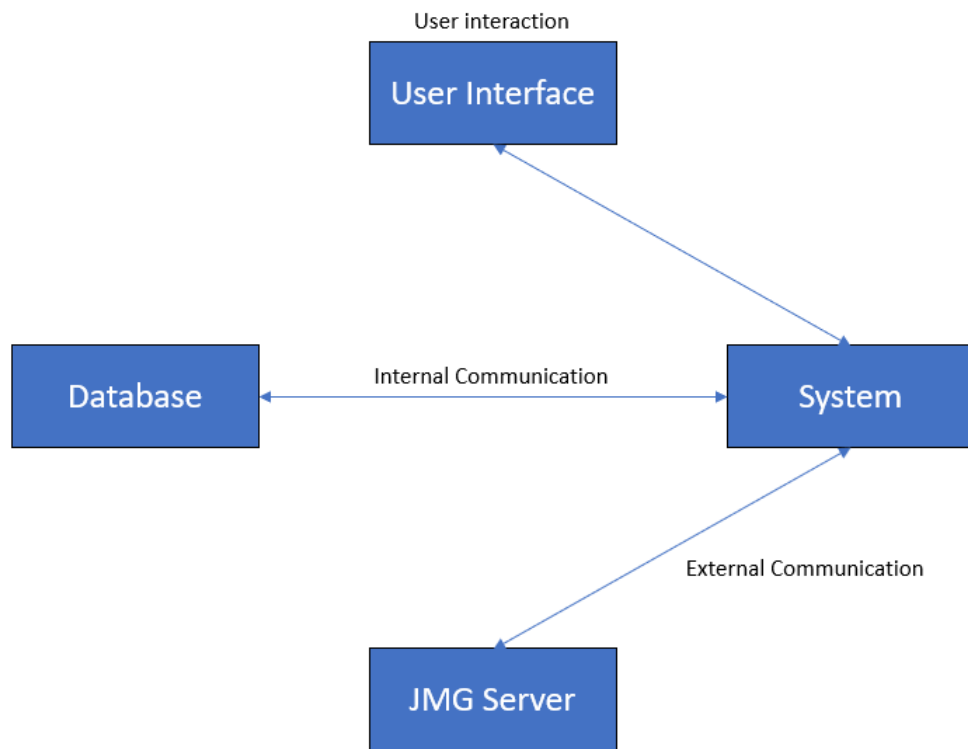


Figure 3-2-a: System Architecture Overview. Four major components are shown above, including the user interface, the system, the database, and the JMG server.

The user interface constitutes the application's front end and will serve 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. The system must also communicate with the JMG Server (Salesforce) in order to request and send data.

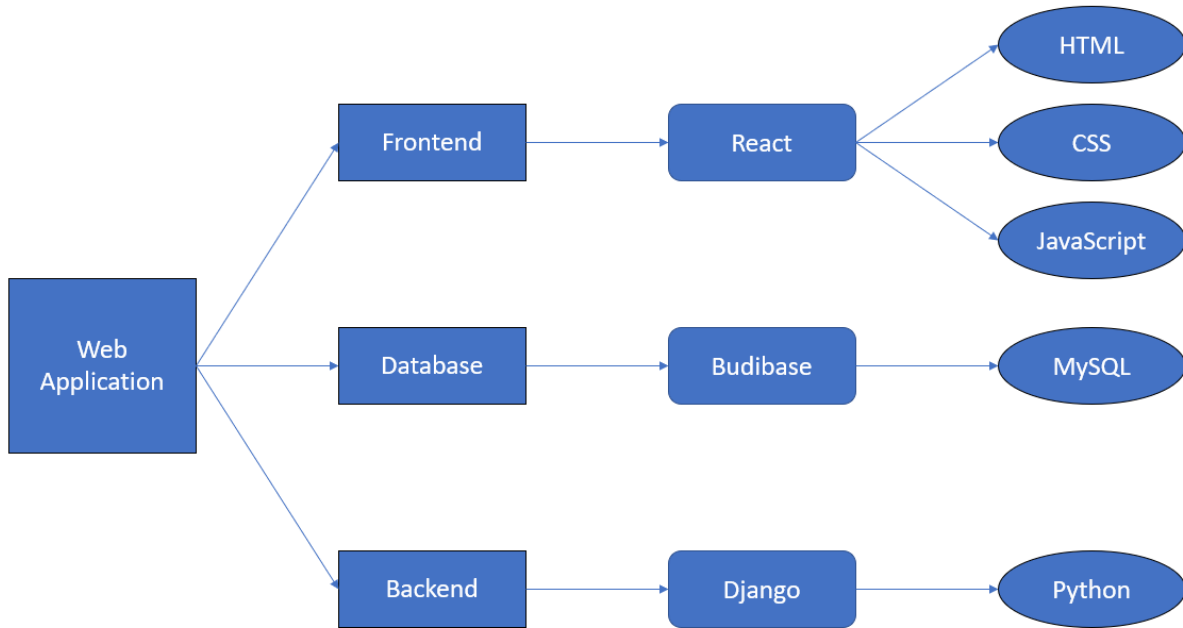


Figure 3-2-b: Technical Architecture Overview. The web application is split into three logical components, with the rectangle shape representing components, the round rectangle shape representing platforms, and the ellipse shape representing programming languages.

Looking at the planned 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 is React. React is among the most popular JavaScript library for building the user interface featuring declarative development view and encapsulated object components. The database component contains and organizes the user data. Our choice for the database platform is Budibase, which is further described in section 3-2-c. 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.

b. Database

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

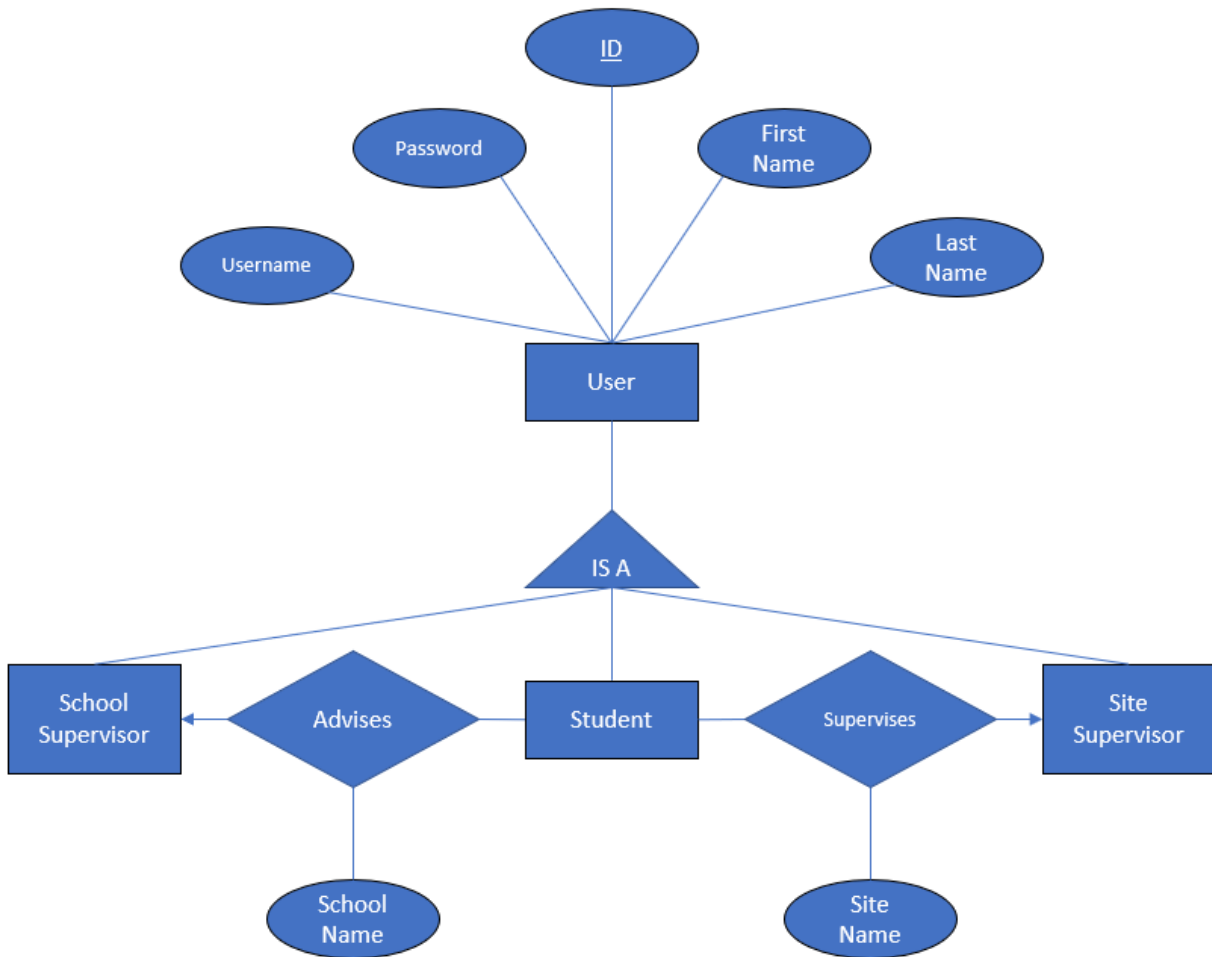


Figure 3-2-c: User ER Model. This model is used to help describe user data and make a distinction between various kinds of users

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.

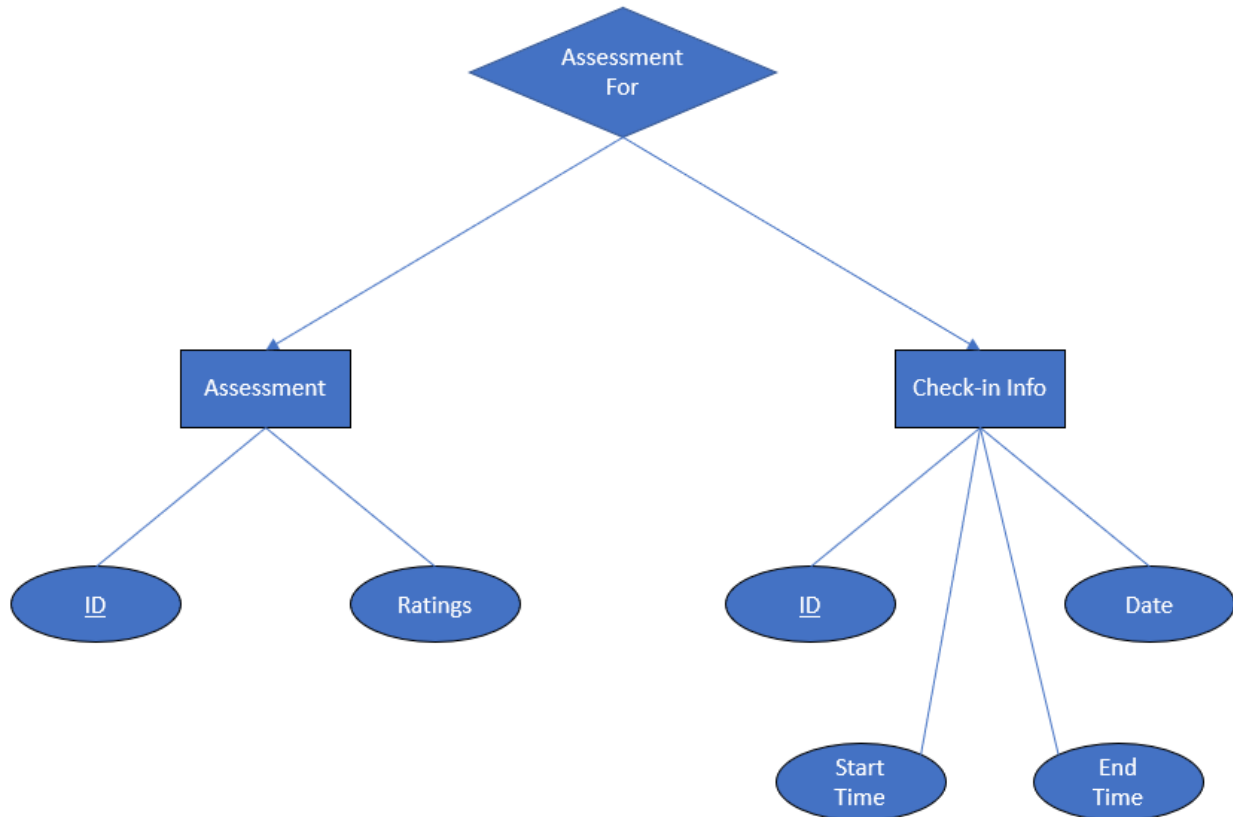


Figure 3-2-d: Check-In Data ER Model. This model contains the data stored for each student check-in

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.

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 a RESTful service's - 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. The 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.

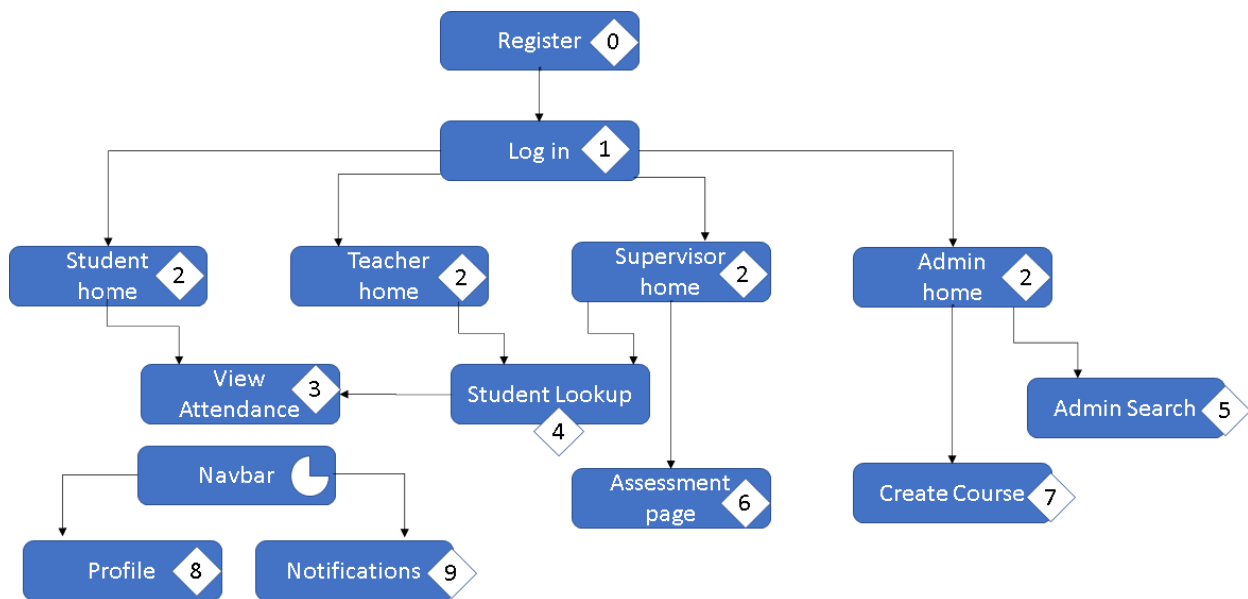


Figure 3-2-e: Navigation flowchart.

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

0

Register

Username

{Username}

Password

Name

First...

Last...

Role

▼ {Role}

Organization

{Organization}

Submit

1

Log in

Username

{Username}

Password

Submit

Figure 3-2-f: Registration and 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-j.), with the added feature for the supervisor to write an assessment of the given student.

Home 7 Profile

2

Home

{Course Title}

Check-in No Check-In detected

View Attendance

{Course Title}

Check-in Checked in Today ✓

View Attendance

...

4

Student lookup

Profile

Search...

{Course} {Name}

Assessment Attendance

Lemon Co. Cave Johnson

Assessment Attendance

States Craft Alexander Hamilton

Assessment Attendance

← {Page #} →

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}

← {Page #} →

Figure 3-2-h: Home, Figure 3-2-i: Student Lookup, and Figure 3-2-j: Attendance.

The final important divergence in site functionality is in the admin account, which gets access to both course creation privileges (figure 3-2-k) and an admin search (figure 3-2-

l.) capable of browsing both users and courses to delete them or view the pages from the perspective of any other user.

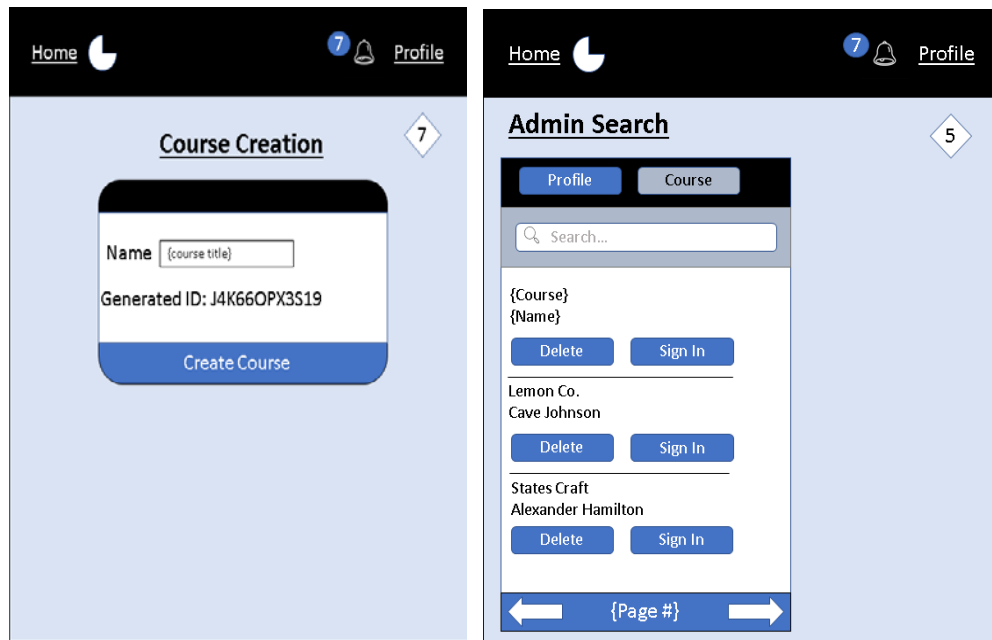


Figure 3-2-k: Course Creation and Figure 3-2-l: Admin Search.

IV. 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 cannot see other student's data. Budibase will hopefully cover 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

Starting winter break, the team plans on starting development of the application. Elijah will begin exploring Budibase and working on developing an implementation of the database. Michael will begin writing up the scripts for the user interface. Xingzhou and Spencer will start looking into any additional backend functionality needed that isn't provided in Budibase through the Django framework. An initial prototype of the application is anticipated to be ready by sometime between January and March of 2022. Subsequent versions of the application will then be rolled out throughout the rest of the Spring 2022 semester.

3. Timeline of Deliverables

The first document created was the System Requirements Specification, which was finished and delivered on the 25th of October 2021. This was followed by the System Design Document, delivered on the 10th of November 2021. A few weeks later on the 29th of November 2021, the User Interface Design Document was finished. The final document to be delivered this semester is this document, the Critical Design Review Document, which will be delivered at the time of writing this on the 13th of December 2021.

4. Team Roles

Xingzhou Luo - Version Control Master

- Created and maintained GitHub repository
- Formatted final versions of all documents
- Lead in system design and architecture

Michael Ferris - Support Manager

- Lead in user interface design

Elijah Caret - Communications Liaison and Team Manager

- Main source of communication between clients and team
- Lead in database design
- Lead in requirements design

Spencer Morse - Additional Support

- Support in requirements design
- Adding additional information for each document such as appendices and introductions

5. 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.

At the end of this process, we want our application to become a component of a system that is "Student centered, results driven."

Appendix A: Document Contributions

- Elijah Caret (50%):
Section I, Section II, Section III, Section IV
- Michael Ferris (30%):
Section II, Section III
- Xingzhou Luo (10%):
Appendix
- Spencer Morse (10%):
Section III

Appendix B: System Requirements Specification

JMG Student Site Check-in Application

System Requirements Specification

Client

Lanet Anthony, Samantha Brink, JMG

Developer



Cyber Cookie

Elijah Caret, Michael Ferris, Xingzhou Luo,
Spencer Morse, Brennan Schatzabel

University of Maine
October 25, 2021
Version 1.1

JMG Student Site Check-In Application
System Requirements Specification

Table of Contents

| | | |
|------|---|----|
| I. | Introduction | 20 |
| | 1. Purpose of the Document | 20 |
| | 2. Purpose of the Product | 20 |
| | 3. Product Scope | 20 |
| | 4. References | 21 |
| II. | Functional Requirements | 21 |
| III. | Non-Functional Requirements | 27 |
| IV. | User Interface | 27 |
| V. | Deliverables | 27 |
| VI. | Open Issues | 28 |
| | Appendix A: Agreement between Customer and Contractor | 29 |
| | Appendix B: Team Review Sign-off | 30 |
| | Appendix C: Document Contributions | 31 |

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.

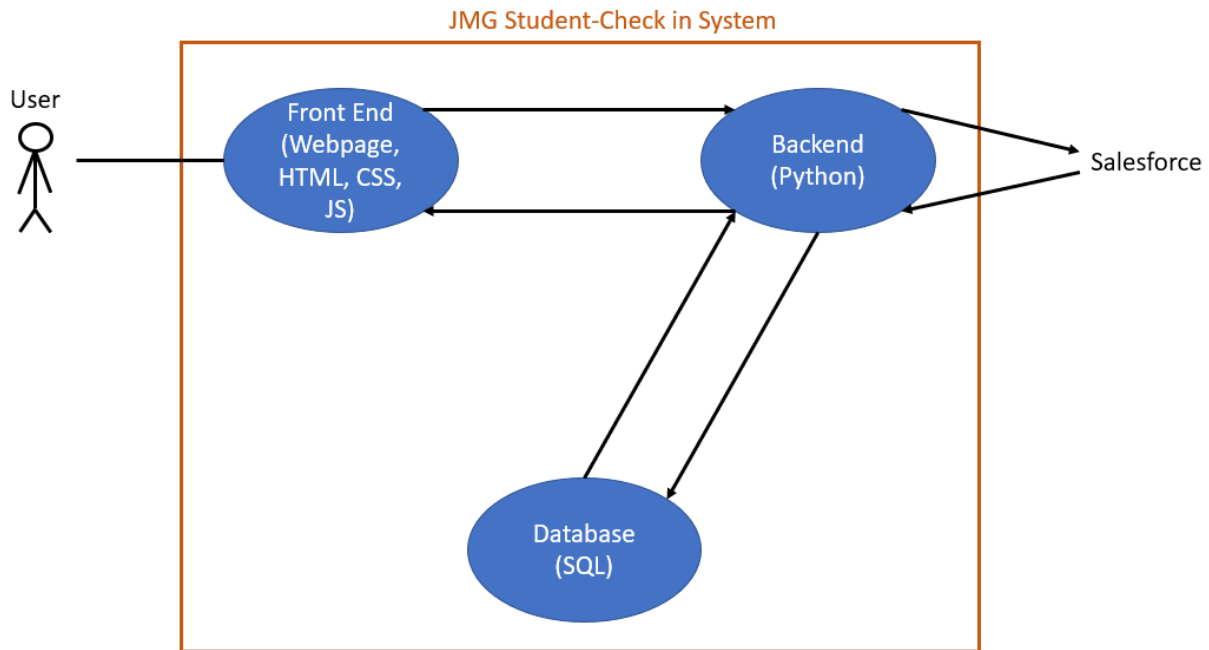


Figure 1: product scope overview

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

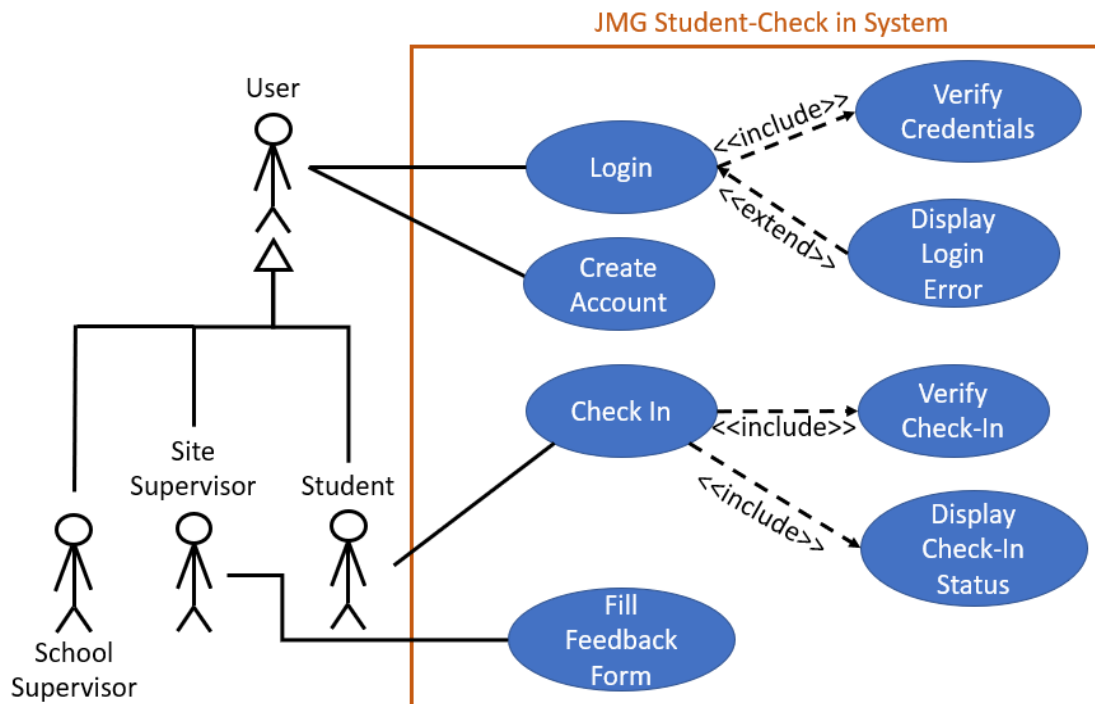


Figure 2.1: use case diagram for use cases 1 - 3, and 6 - 9

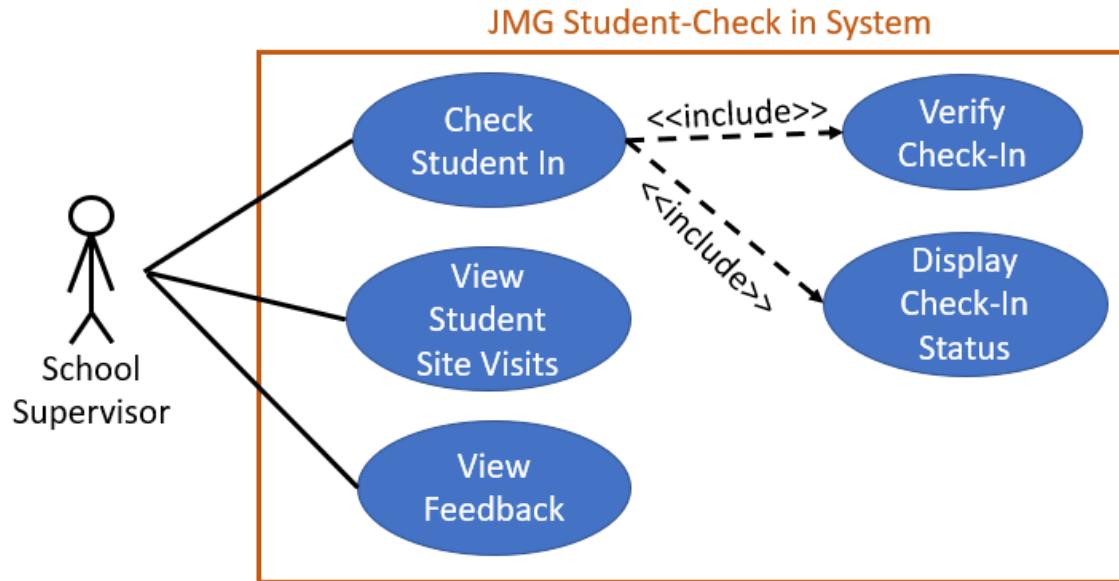


Figure 2.2: use case diagram for use cases 10 - 13

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:

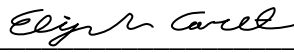
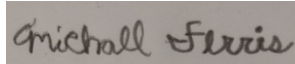
- Process of determining when a student can no longer check themselves in, especially during unexpected site visits.

Appendix A: Agreement between Customer and Contractor

By signing this document, all parties agree that this is a complete list of requirements and deliverables for the JMG Student Check-in Site application, apart from the current open issues (addressed in Section VI). The tests described in this document (under sections II and III) will be the metric by which the product is deemed complete. Once all requirements are complete and verified by said tests, the application will be finished and ready for use by JMG.

In the case that requirements or any other information in this document need to change for the contract to be fulfilled, the following procedure will be followed: The party that believes a change is necessary shall contact the other party, explaining the situation. A meeting between the two parties will be held to discuss the problem and possible solutions. Once an agreement has been reached, modifications to this document will be made to reflect it, and all members of each party will sign the new document, which will then replace this one.

Team Members:

| | |
|-----------------------------|--|
| Name: <u>Elijah Caret</u> | Signature: <u></u> |
| Name: <u>Michael Ferris</u> | Signature: <u></u> |
| Name: <u>Xingzhou Luo</u> | Signature: <u>Xingzhou Luo</u> |
| Name: <u>Spencer Morse</u> | Signature: <u>Spencer Morse</u> |

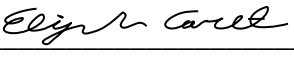
Customers:

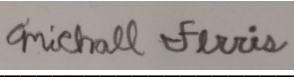
| | |
|-----------------------------|--|
| Name: <u>Samantha Brink</u> | Signature: <u></u> |
|-----------------------------|--|

Appendix B: Team Review Sign-off

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

Team Members:

Name: Elijah Caret Signature:  Date: Oct. 25, 2021

Name: Michael Ferris Signature:  Date: Oct. 25, 2021

Name: Xingzhou Luo Signature: Xingzhou Luo Date: Oct. 25, 2021

Name: Spencer Morse Signature: Spencer Morse Date: Oct. 25, 2021

Appendix C: Document Contributions

- Elijah Caret (40%):
Introduction, Functional Requirements, Non-Functional Requirements,
Deliverables, Open Issues, Appendices
- Michael Ferris (15%):
Introduction, Appendices
- Xingzhou Luo (25%):
Introduction, Functional Requirements, Non-Functional Requirements,
Appendices
- Spencer Morse (15%):
Open Issues, Appendices
- Brennan Schatzabel (5%):
Introduction

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 1.1



JMG Student Site Check-In Application System Design Document

Table of Contents

| | | |
|------|---|----|
| I. | Introduction | 35 |
| | 5. Purpose of the Document | 35 |
| | 6. References | 35 |
| II. | System Architecture | 36 |
| | 1. Architectural Design | 36 |
| | 2. Decomposition Description | 38 |
| III. | Persistent Data Design | 38 |
| | 1. Database Descriptions | 39 |
| | 2. File Descriptions | 41 |
| IV. | Requirements Matrix | 41 |
| | Appendix A: Agreement between Customer and Contractor | 43 |
| | Appendix B: Team Review Sign-off | 44 |
| | Appendix C: Document Contributions | 45 |

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.

5. 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.

6. 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.

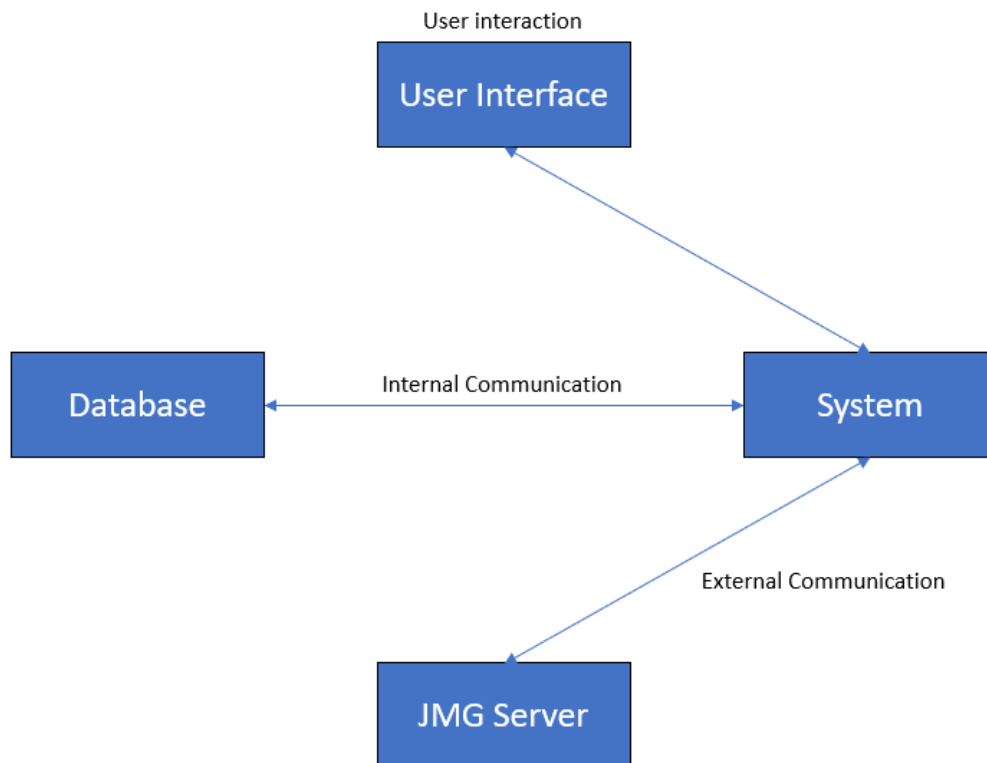


Figure 2.1: System Architecture Overview – Four major components are shown above, including the user interface, the system, the database, and the JMG server.

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.

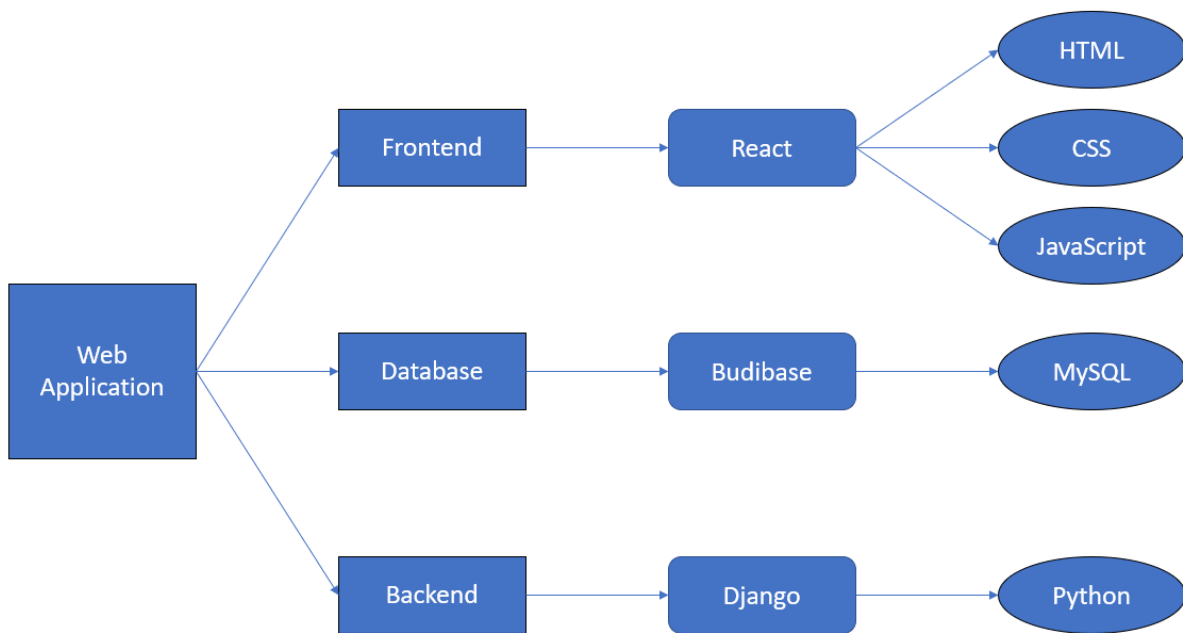


Figure 2.2: Technical Architecture Overview – The web application is split into three logical components, with the rectangle shape representing components, the squircle shape representing platforms, ad the ellipse shape representing programming languages.

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

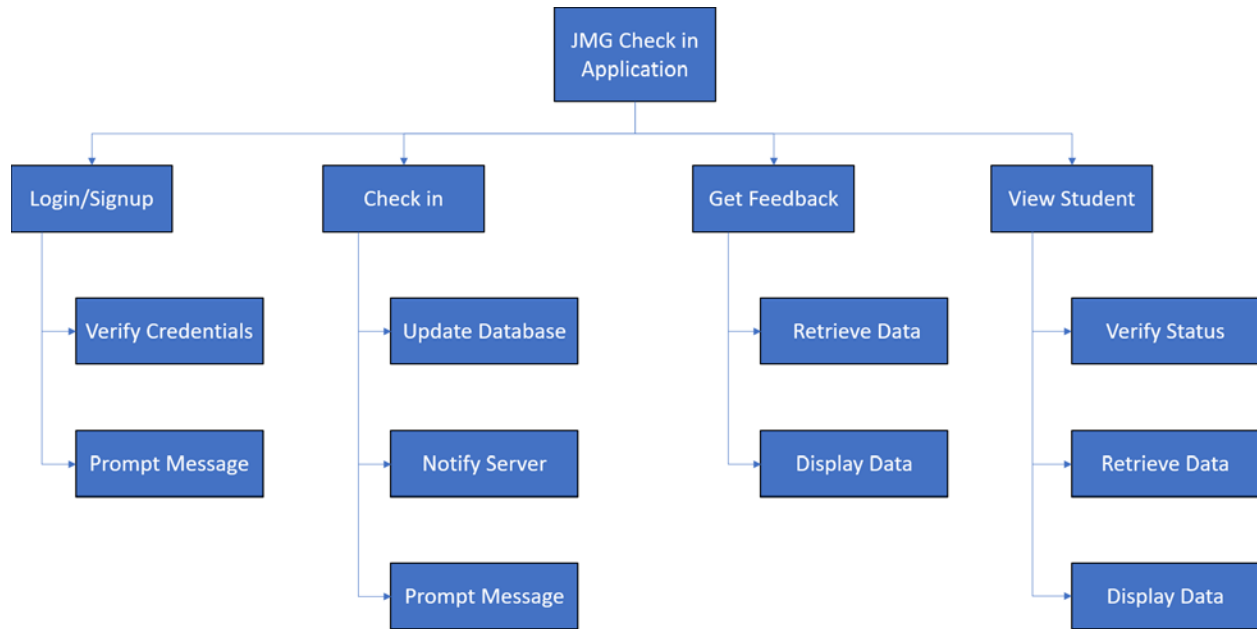


Figure 2.3: Structural Decomposition (Hierarchy) Overview – Four major functions of the web application is shown above, including user login/signup, student check-in, student getting feedback, and supervisor viewing student record.

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.

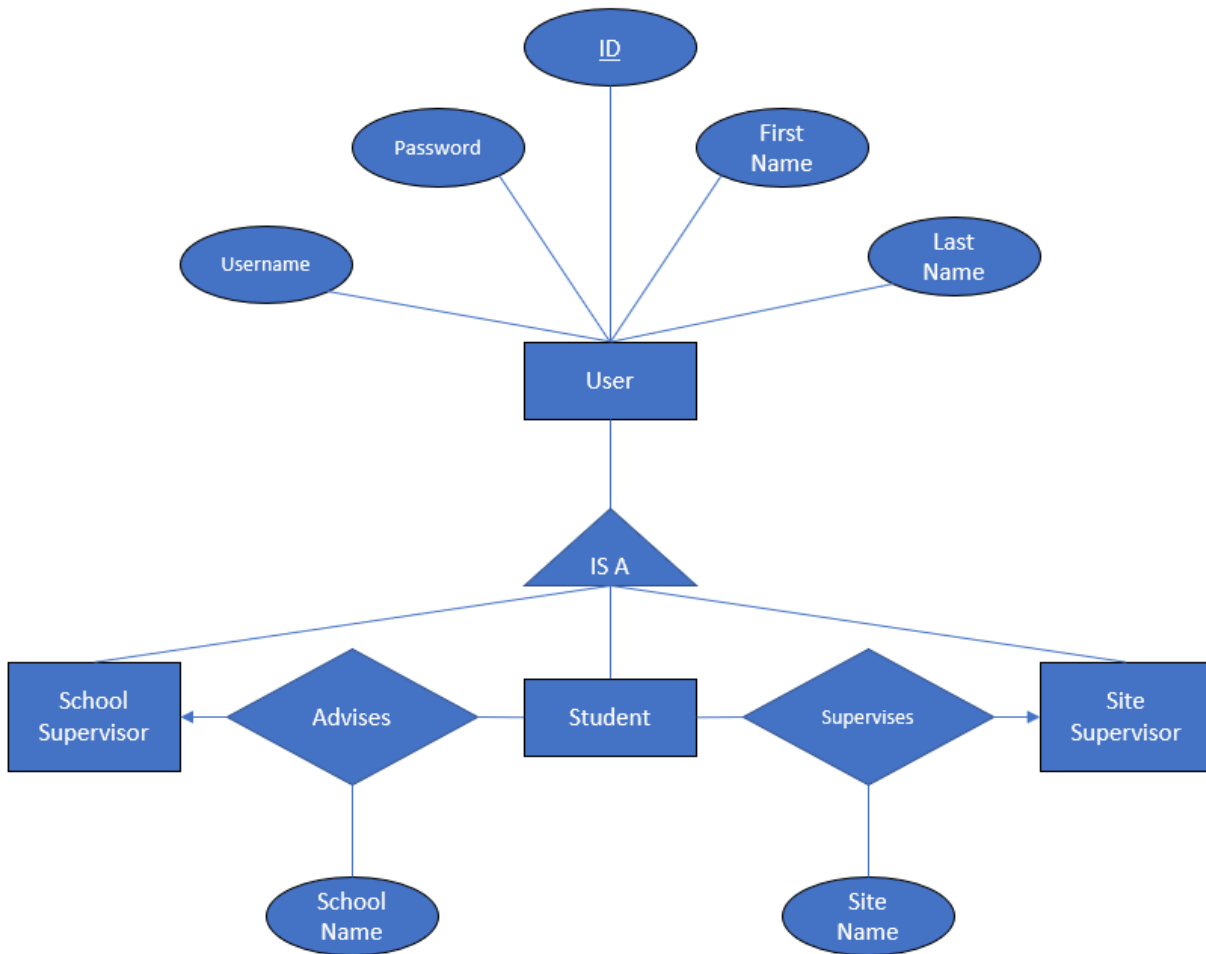


Figure 3.1: ER Model for User Table – Each user will have a username, password, ID number, first name, and last name. The student, school supervisor, and site supervisor are all users linked to each other via the supervises and advises relations.

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.

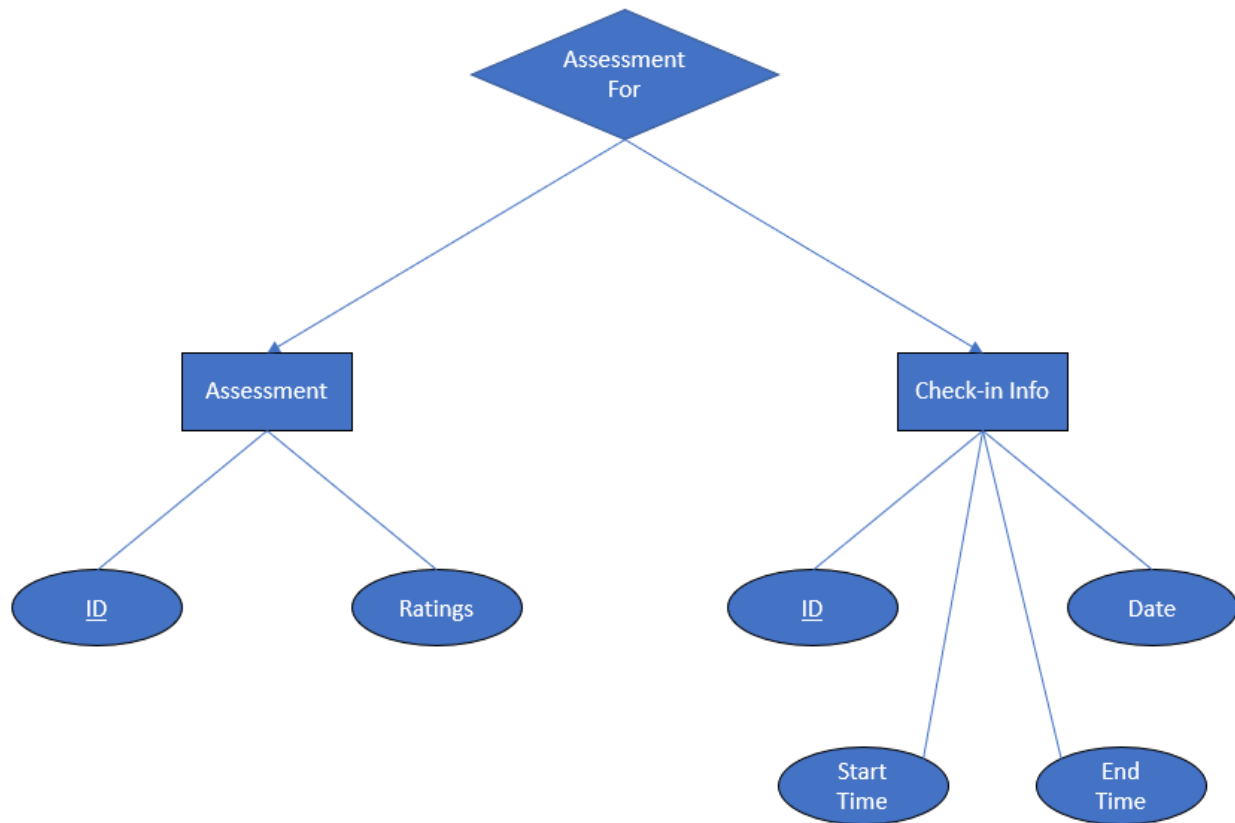


Figure 3.2: Site Check-in Model, Assessment Table, and Relationship with Check-in – The student is also linked to check-ins, which have a start time, end time, date, and id. Each check in will have an assessment to go along with it.

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 | | | | | X | X | X | |

| | | | | | | | | |
|---------------------------|--|--|--|--|---|---|---|---|
| of Student Check-in | | | | | | | | |
| 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 A: Agreement between Customer and Contractor

By signing this document, all parties agree that this is a complete list of system architecture and design for the JMG Student Check-in Site application.

In the case that system architecture and design or any other information in this document need to change for the contract to be fulfilled, the following procedure will be followed: The party that believes a change is necessary shall contact the other party, explaining the situation. A meeting between the two parties will be held to discuss the problem and possible solutions. Once an agreement has been reached, modifications to this document will be made to reflect it, and all members of each party will sign the new document, which will then replace this one.

Team Members:

| | |
|-----------------------------|---|
| Name: <u>Elijah Caret</u> | Signature: <u><i>Elijah Caret</i></u> |
| Name: <u>Michael Ferris</u> | Signature: <u><i>Michael Ferris</i></u> |
| Name: <u>Xingzhou Luo</u> | Signature: <u><i>Xingzhou Luo</i></u> |
| Name: <u>Spencer Morse</u> | Signature: <u><i>Spencer Morse</i></u> |

Customers:

| | |
|-----------------------------|---|
| Name: <u>Samantha Brink</u> | Signature: <u><i>Samantha Brink</i></u> |
| Name: <u>Samantha Brink</u> | Signature: <u><i>Lanet Anthony</i></u> |

Appendix B: Team Review Sign-off

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

Team Members:

Name: Elijah Caret Signature: *Elijah Caret* Date: Nov. 10, 2021

Name: Michael Ferris Signature: *Michael Ferris* Date: Nov. 10, 2021

Name: Xingzhou Luo Signature: *Xingzhou Luo* Date: Nov. 10, 2021

Name: Spencer Morse Signature: *Spencer Morse* Date: Nov. 10, 2021

Appendix C: Document Contributions

- Elijah Caret (35%):
Section I, Section III, Appendix A, Appendix B, Appendix C
- Michael Ferris (5%):
Appendix A, Appendix B
- Xingzhou Luo (35%):
Section I, Section II, Appendix A, Appendix B, Appendix C
- Spencer Morse (25%):
Section I, Section IV, Appendix A, Appendix B, Appendix C

Appendix D: User Interface Design Document

JMG Student Site Check-in Application

User Interface Design Document

Client

Lanet Anthony, Samantha Brink, JMG

Developer



Cyber Cookie

Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
November 29, 2021
Version 1.1



JMG Student Site Check-In Application
User Interface Design Document

Table of Contents

| | | |
|------|---|----|
| I. | Introduction | 49 |
| | 7. Purpose of the Document | 49 |
| | 8. References | 49 |
| II. | User Interface Standards | 49 |
| III. | User Interface Walkthrough | 52 |
| IV. | Data Validation | 57 |
| | Appendix A: Agreement between Customer and Contractor | 59 |
| | Appendix B: Team Review Sign-off | 60 |
| | Appendix C: Document Contributions | 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.

7. 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.

8. 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.

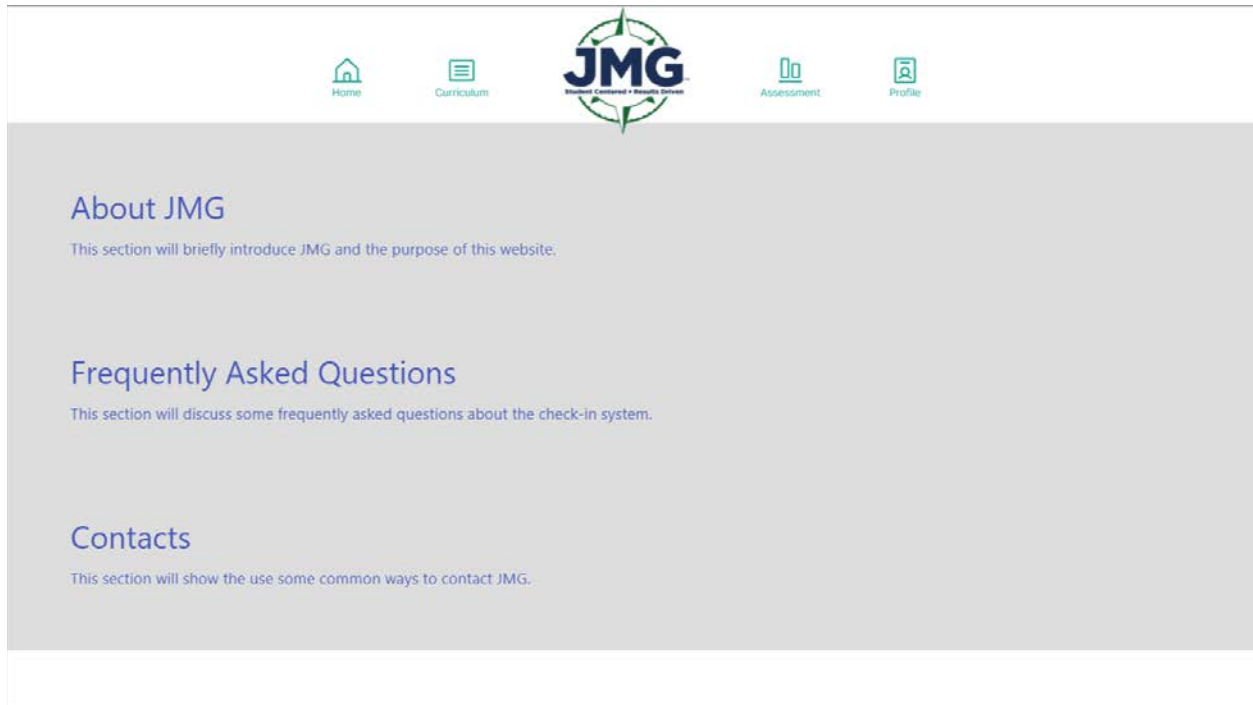


Figure 2.1: The home page view of the system. There are three sections, including “About JMG”, “Frequently Asked Questions”, and “Contacts”.

| | 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 2.2: The curriculum page view. Available courses and their schedules will be listed here.

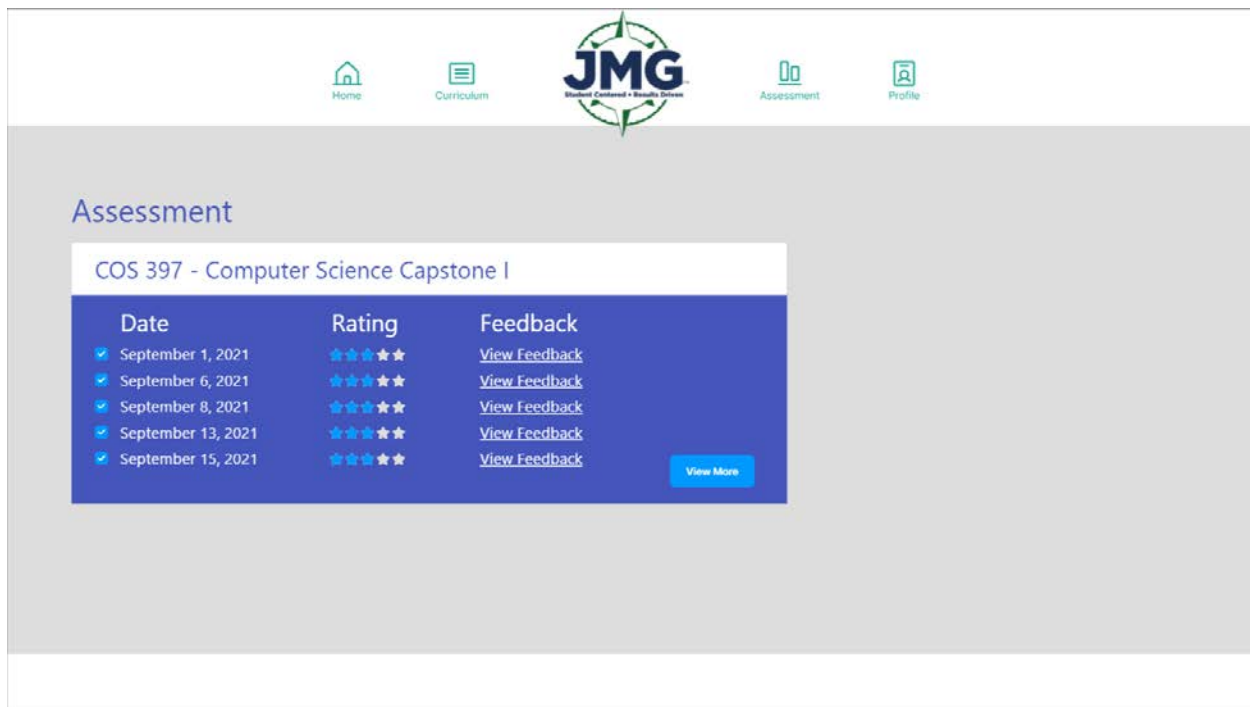


Figure 2.3: The assessment page view. The user can access to their course assessment form here.

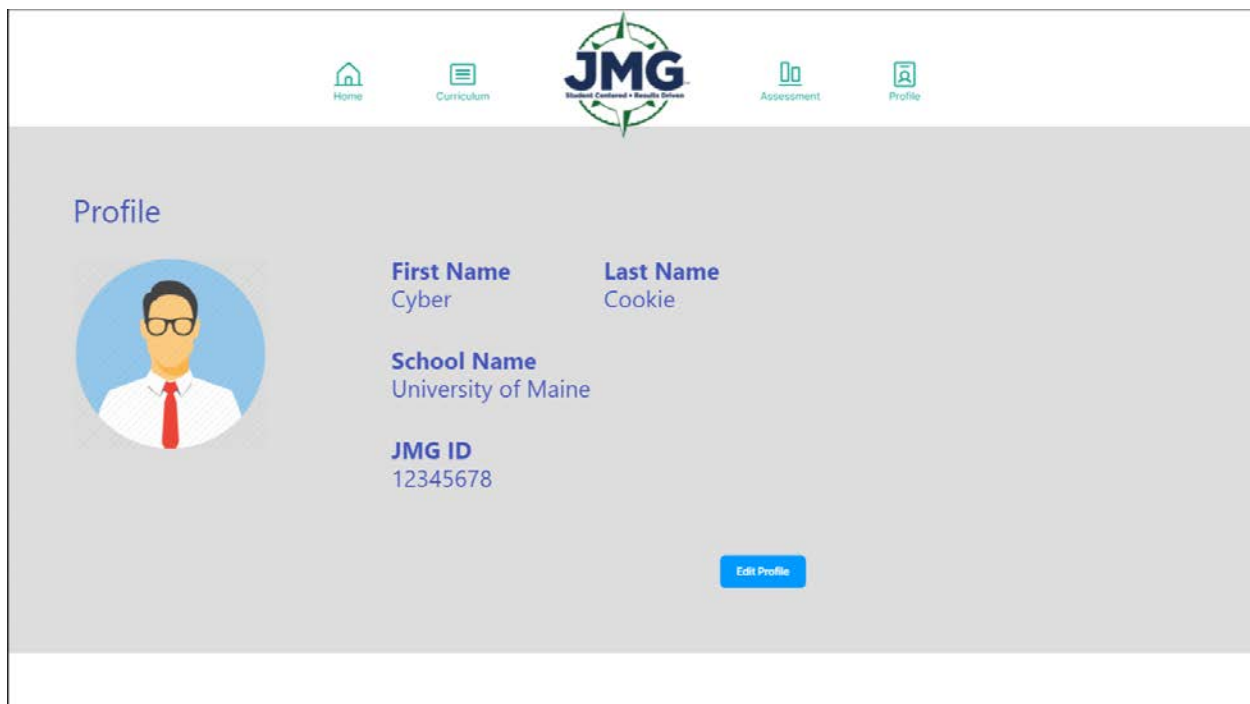


Figure 2.4: The profile page. The user can view and edit its full name, school name, and ID here.

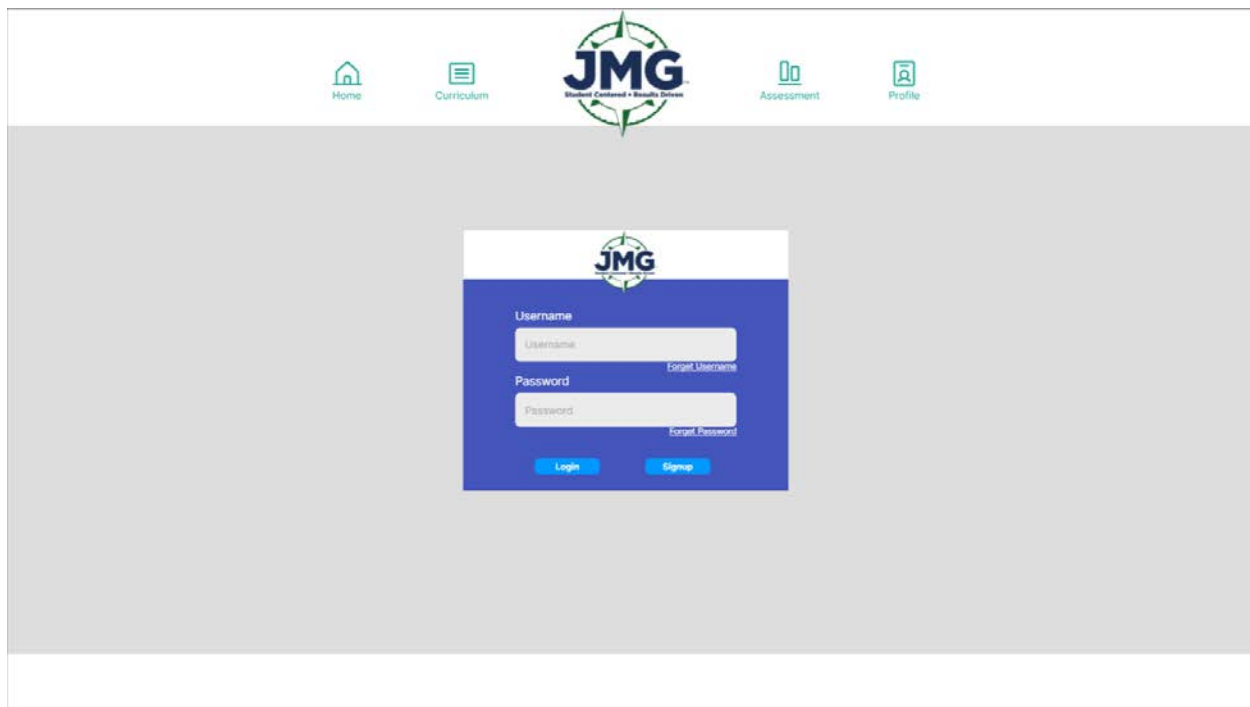


Figure 2.5: The credential page view of the system. A user must login or signup before using other features on the website.

III. Persistent Data Design

Provide a diagram that illustrates how the user will navigate from one screen to another (I call this a “navigation diagram”). Label each symbol that represents a screen so that you can reference the screens, if necessary, later in the document. Give a brief description of what the diagram as a whole represents.

Next, guide the reader through a series of screenshots of all system screens. (You do not need to include error and confirmation messages/pop-ups.) Give the screen shots figure numbers and labels that match those in the navigation diagram. Refer to the figure numbers in the text of the walkthrough. Explain what the reader is seeing in each screen shot: the major screen areas, menus, what each button does, how to navigate to the next screen or return to the previous, etc. Note that if a feature has been standardized (*e.g.*, how to return to the previous screen) and explained in Section 2, you do not need to repeat it here.

Screen shots should not be handwritten but may be drawn using any tool you wish (*e.g.*, Visual Basic, PowerPoint, a drawing program, etc.). At this point in the process, they are a best approximation as to what the user interface will look like in both layout and annotation.

Navigation Diagram

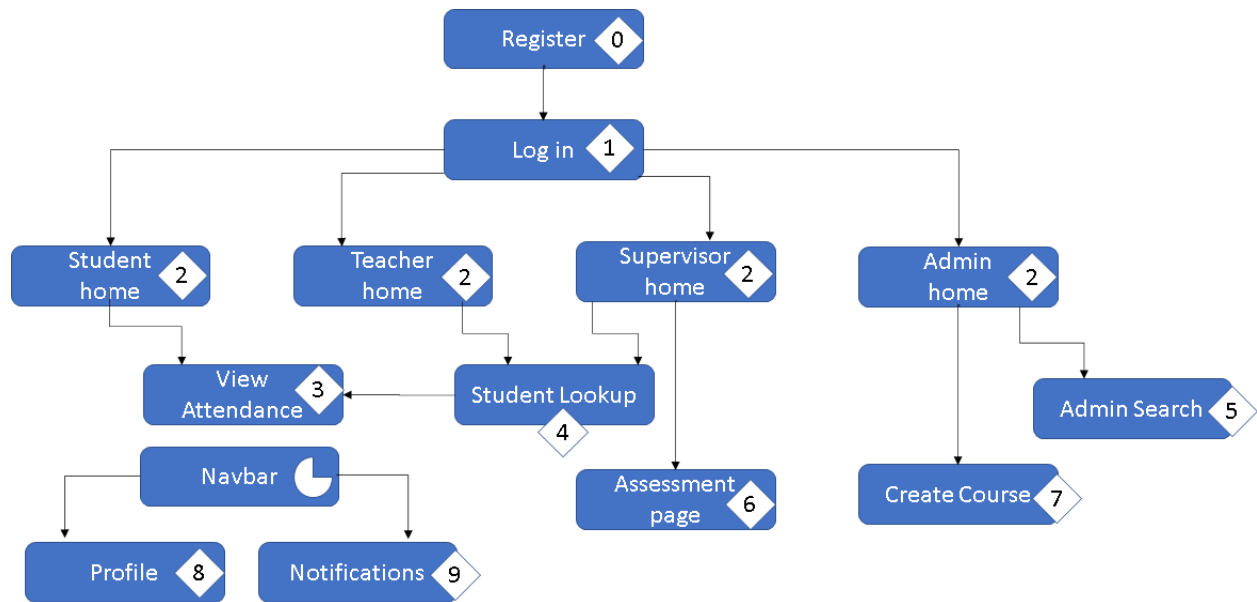


Figure 3.1: Site Navigation Flow Chart

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 3.2: Registration and Figure 3.3: Login

Home  7  Profile

Home

2

{Course Title}

Check-In No Check-In detected



View Attendance

{Course Title}

Check-In Checked in Today ✓

View Attendance

...

Home  7  Profile

Attendance

3

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}

← {Page #} →

Figure 3.4: Home and Figure 3.5: Attendance

54

Home 7 Profile

Student lookup 4

Profile

Search...

| {Course} | {Name} | Assessment | Attendance |
|--------------|--------------------|------------|------------|
| Lemon Co. | Cave Johnson | Assessment | Attendance |
| States Craft | Alexander Hamilton | Assessment | Attendance |

← {Page #} →

Figure 3.6: Student Lookup and Figure 3.7: Admin Search

Home 7 Profile

Admin Search 5

Profile **Course**

Search...

| {Course} | {Name} | Delete | Sign In |
|--------------|--------------------|--------|---------|
| Lemon Co. | Cave Johnson | Delete | Sign In |
| States Craft | Alexander Hamilton | Delete | Sign In |

← {Page #} →

Home 7 Profile

Assessment 6

{Course Title}

{Student} reported:

Date: 11/16/21

At: 8:37 AM

Confirm **Deny**

Review Criteria #1

☒ ☐ ☐ ☐ ☐

...

Free Write

Apprentice seemed a little tired.
Overall good job.
Keep up the good work.

Overall 4.2/5

Submit

Home 7 Profile

Course Creation 7

Name {course title}

Generated ID: J4K66OPX3S19

Create Course

Figure 3.8: Assessment and Figure 3.9: Course Creation

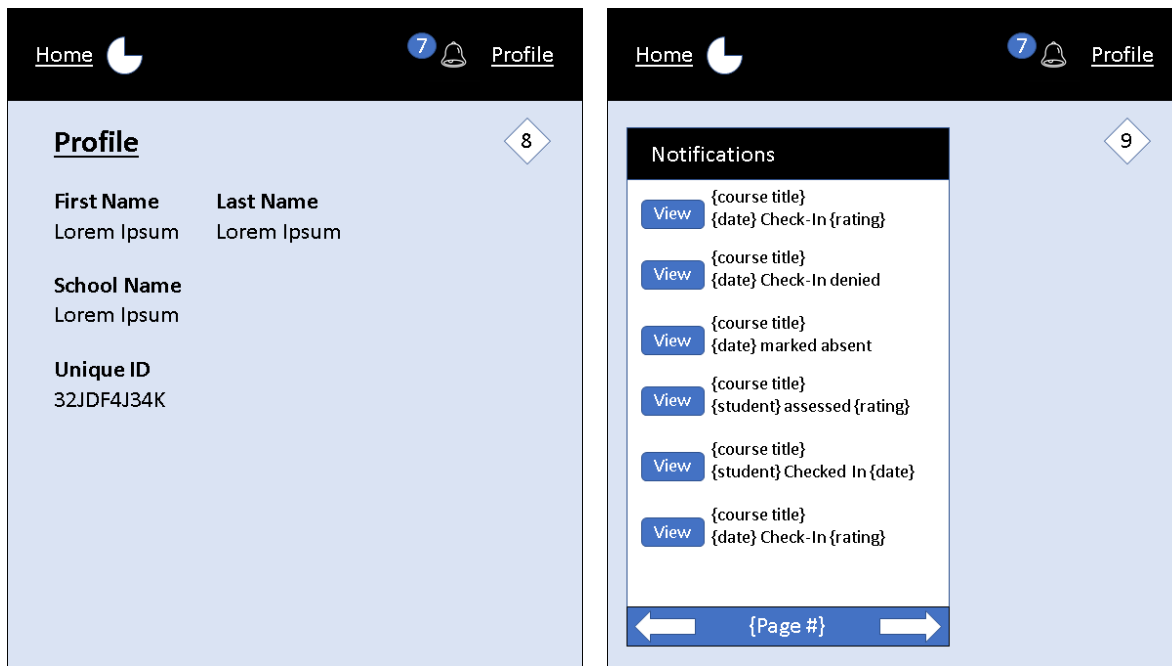


Figure 3.10: Profile and Figure 3.11: Notifications

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” |
| Check In: Begin Time | String | Max Hour: 12 Max Minute: 59 | “HH:MM (meridian)” or “HH:MM(meridian)” |
| Check In: End Time | String | Max Hour: 12 Max Minute: 59 | “HH:MM (meridian)” or “HH:MM(meridian)” |
| 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 4: 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 A: Agreement between Customer and Contractor

By signing this document, all parties agree that this is a complete list of user interface features for the JMG Student Check-in Site application.

In the case that user interface features or any other information in this document need to change for the contract to be fulfilled, the following procedure will be followed: The party that believes a change is necessary shall contact the other party, explaining the situation. A meeting between the two parties will be held to discuss the problem and possible solutions. Once an agreement has been reached, modifications to this document will be made to reflect it, and all members of each party will sign the new document, which will then replace this one.

Team Members:

| | |
|-----------------------------|---|
| Name: <u>Elijah Caret</u> | Signature: <u><i>Elijah Caret</i></u> |
| Name: <u>Michael Ferris</u> | Signature: <u><i>Michael Ferris</i></u> |
| Name: <u>Xingzhou Luo</u> | Signature: <u><i>Xingzhou Luo</i></u> |
| Name: <u>Spencer Morse</u> | Signature: <u><i>Spencer Morse</i></u> |

Customers:

| | |
|-----------------------------|---|
| Name: <u>Samantha Brink</u> | Signature: <u><i>Samantha Brink</i></u> |
| Name: <u>Samantha Brink</u> | Signature: <u><i>Lanet Anthony</i></u> |

Appendix B: Team Review Sign-off

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

Team Members:

Name: Elijah Caret Signature: *Elijah Caret* Date: Nov. 29, 2021

Name: Michael Ferris Signature: *Michael Ferris* Date: Nov. 29, 2021

Name: Xingzhou Luo Signature: *Xingzhou Luo* Date: Nov. 29, 2021

Name: Spencer Morse Signature: *Spencer Morse* Date: Nov. 29, 2021

Appendix C: Document Contributions

- Elijah Caret (25%):
Section IV, Appendix A, Appendix B, Appendix C
- Michael Ferris (35%):
Section III, Appendix A, Appendix B, Appendix C
- Xingzhou Luo (30%):
Section II, Appendix A, Appendix B, Appendix C
- Spencer Morse (10%):
Section I, Appendix A, Appendix B, Appendix C