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 1.2

JMG Student Site Check-In Application
System Design Document

Table of Contents

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)

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

React. *A JavaScript library for building user interfaces (2021).* Retrieved November 10, 2021, from
React – A JavaScript library for building user interfaces (reactjs.org)

Budibase. *Build internal tools, the easy way (2021).* Retrieved November 10, 2021, from
Budibase | 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)

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.

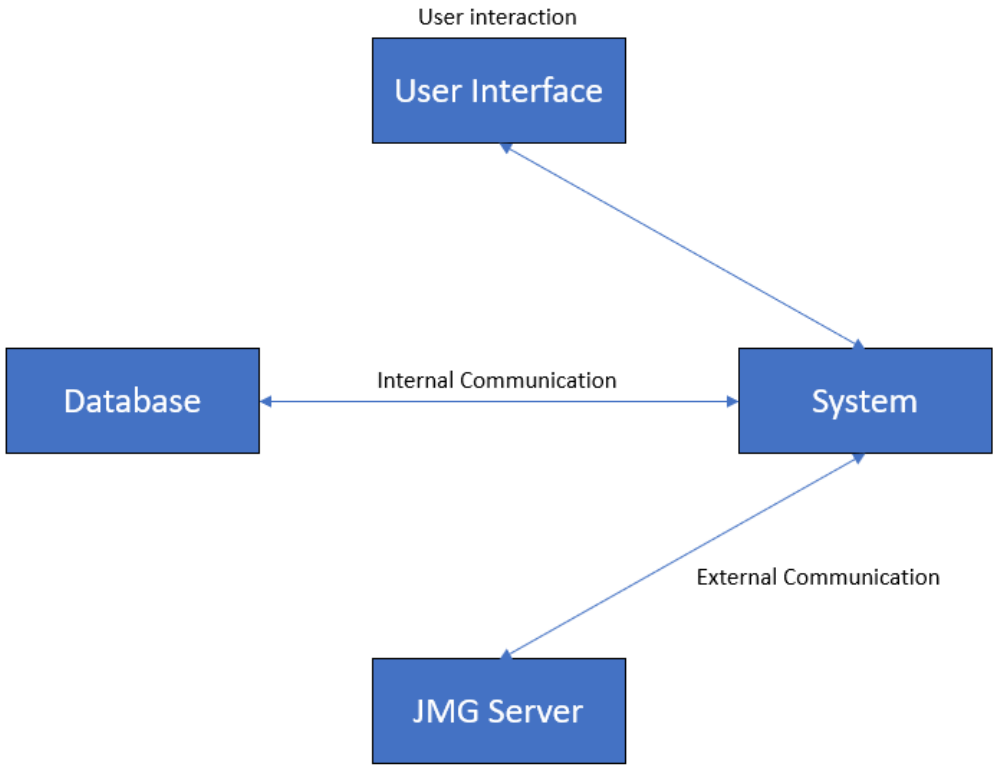


Figure 2.1: System Architecture Overview – Four major components are shown above, including the user interface, the system, the database, and the JMG server.

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.

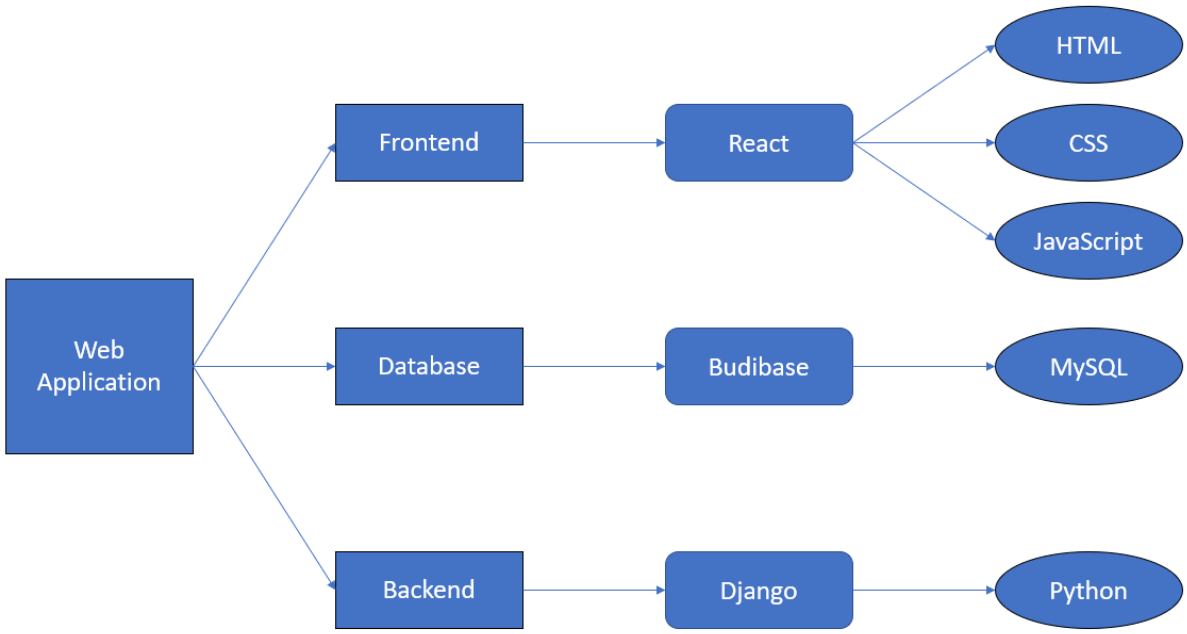


Figure 2.2: Technical Architecture Overview – The web application is split into three logical components, with the rectangle shape representing components, the squircle shape representing platforms, ad the ellipse shape representing programming languages.

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

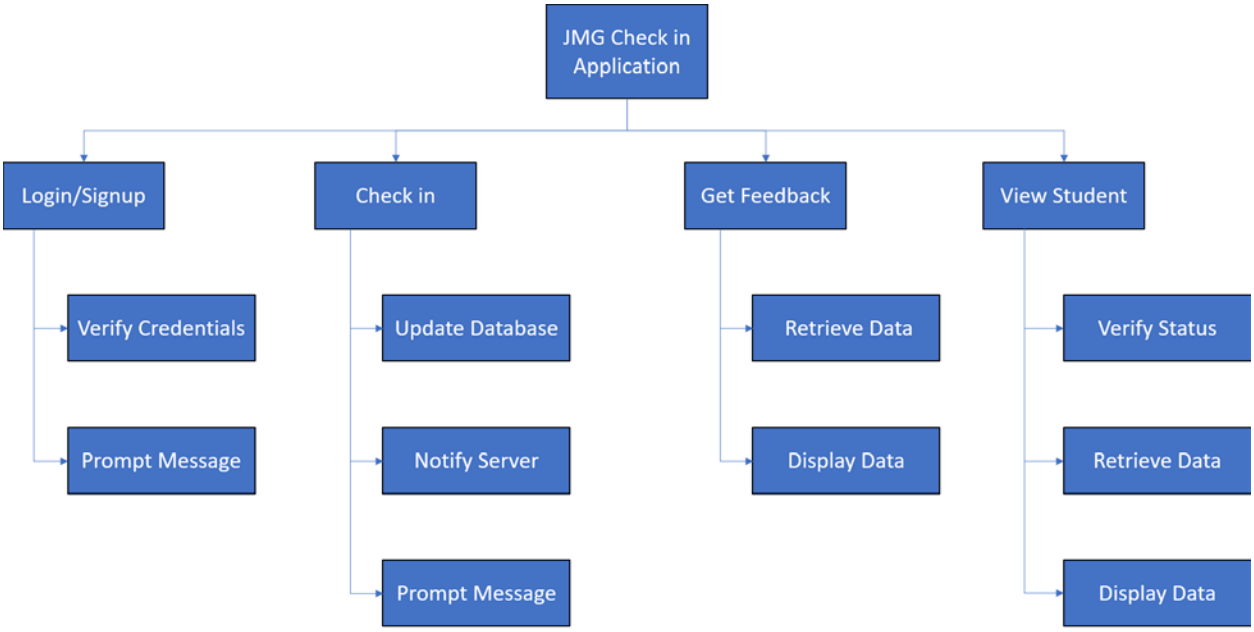


Figure 2.3: Structural Decomposition (Hierarchy) Overview – Four major functions of the web application is shown above, including user login/signup, student check-in, student getting feedback, and supervisor viewing student record.

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.

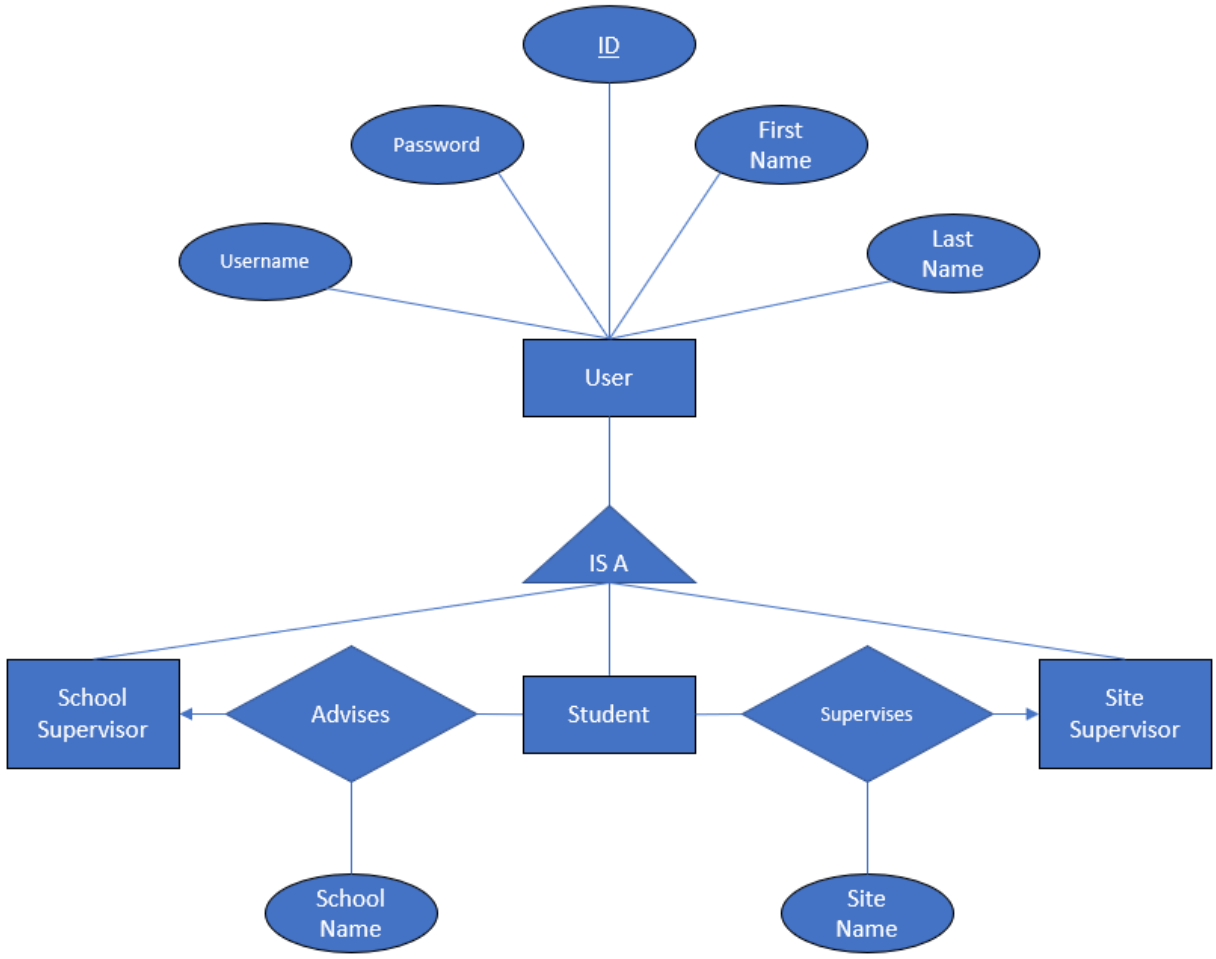


Figure 3.1: ER Model for User Table – Each user will have a username, password, ID number, first name, and last name. The student, school supervisor, and site supervisor are all users linked to each other via the supervises and advises relations.

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.

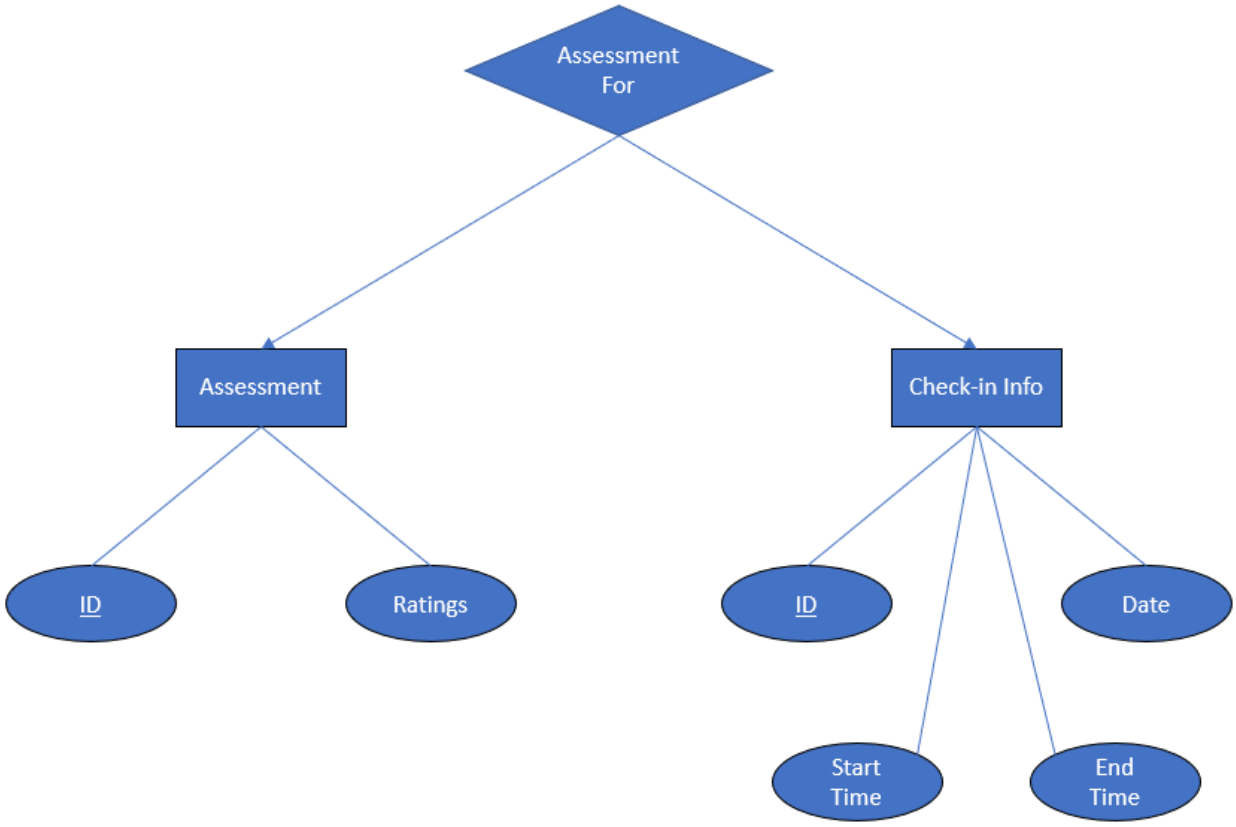


Figure 3.2: Site Check-in Model, Assessment Table, and Relationship with Check-in – The student is also linked to check-ins, which have a start time, end time, date, and id. Each check in will have an assessment to go along with it.

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	

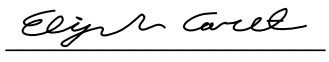
of Student Check-in								
Feedback Form							X	X
View Feedback							X	X
Supervisor Check-in					X		X	
Supervisor Check-in					X	X	X	
View Student Visits						X	X	

Appendix A: Agreement between Customer and Contractor

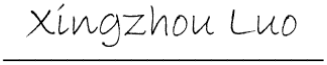
By signing this document, all parties agree that this is a complete list of system architecture and design for the JMG Student Check-in Site application.

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:

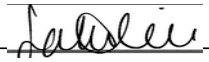
Name: Elijah Caret Signature: 

Name: Michael Ferris Signature: 

Name: Xingzhou Luo Signature: 

Name: Spencer Morse Signature: 

Customers:

Name: Samantha Brink Signature: 

Name: Lanet Anthony Signature: 

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