JMG Student Site Check-in Application

Code Inspection Report

Client

JMG
Lanet Anthony, Samantha Brink

Developer

Cyber Cookie
Elijah Caret, Michael Ferris,
Xingzhou Luo, Spencer Morse

University of Maine
March 9, 2022
Version 1.0

Cyber Cookie

JMG Student Site Check-in Application Design Inspection Report

Table of Contents

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 date 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.

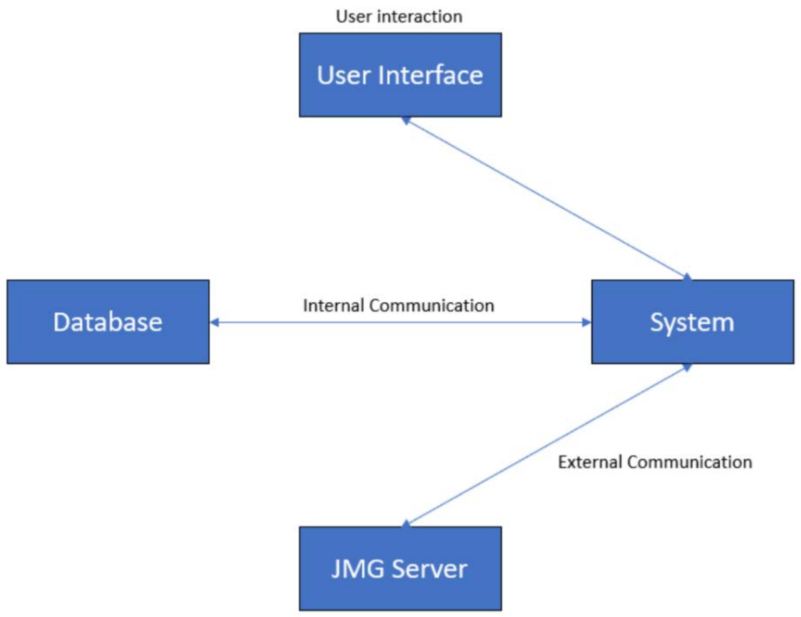

Figure 3.1 - System Architecture Overview.

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.

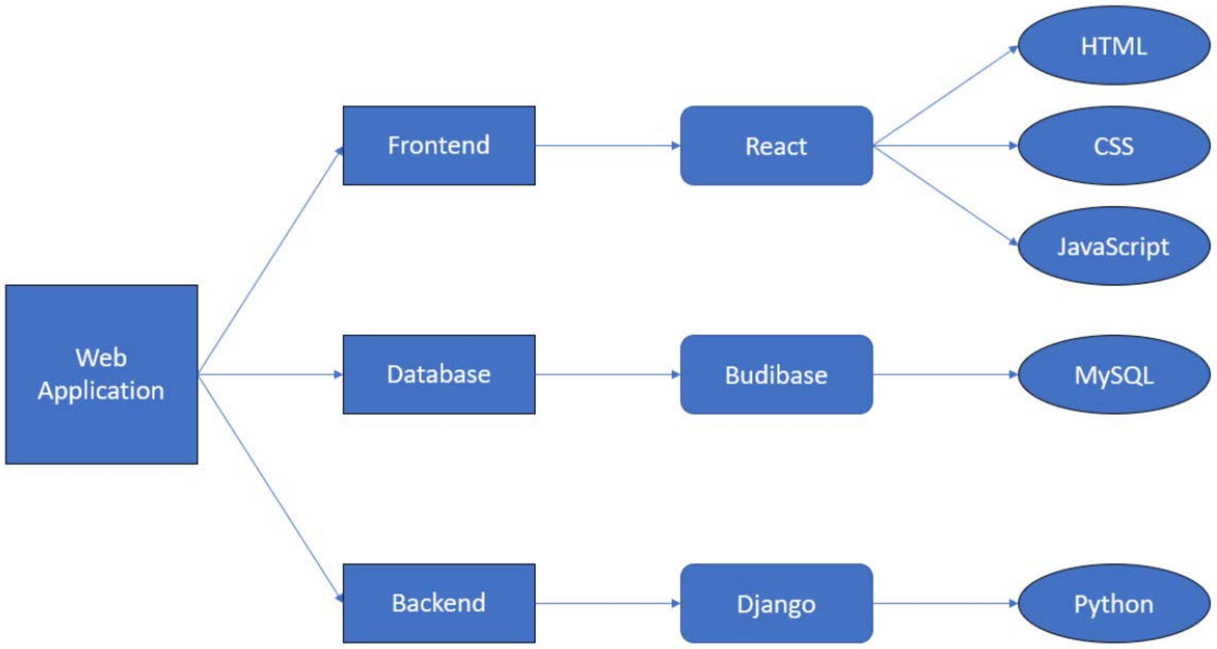

Figure 3.2 - Technical Architecture Overview.

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”.

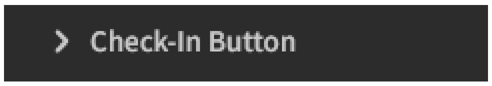

Figure A.1 - Example label of a button: “button” is included in its label.

- 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.

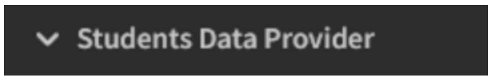

Figure A.2 - Data Provider label: “Students” indicates the table it is drawing data from.

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 B: 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: [illegible]	Date: **3/08/2022**
Name: **Michael Ferris**	Signature: Spencer Morse	Date: **3/08/2022**
Name: **Xingzhou Luo**	Signature: Xingzhou Luo	Date: **3/08/2022**
Name: **Spencer Morse**	Signature: Michael Ferris	Date: **3/08/2022**

Customers:

Name: **Samantha Brink**	Signature: Samantha Brink	Date: **03/09/2022**
Name: **Lanet Anthony**	Signature: Lanet Anthony	Date: **03/09/2022**

Appendix C: Team Review Sign Off

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

Name: **Elijah Caret**	Signature: [illegible]	Date: **3/08/2022**
Name: **Michael Ferris**	Signature: Spencer Morse	Date: **3/08/2022**
Name: **Xingzhou Luo**	Signature: Xingzhou Luo	Date: **3/08/2022**
Name: **Spencer Morse**	Signature: Michael Ferris	Date: **3/08/2022**

Appendix D: Document Contributions

- **Elijah Caret** - 30%
  Section 1.1, Section 1.2, Section 2, Section 4, Appendix A, Appendix B, Appendix C, and Appendix D

- **Michael Ferris** - 20%
  Section 4, Appendix B, Appendix C, and Appendix D

- **Xingzhou Luo** - 30%
  Section 1.1, Section 1.2, Section 2, Section 3, Appendix A, Appendix B, Appendix C, and Appendix D

- **Spencer Morse** - 20%
  Section 1.3, Section 1.4, Appendix B, Appendix C, and Appendix D