

**Avalanche Innovation Team:**

Nishan Subba - Database Administrator

Aman Gurung - Data/Business Analyst I

Chiagoziem Igwe - Data/Business Analyst II

Dhulfiqar Hussein - Lead Developer I, Full Stack

Harshitha Sayini - Developer II, Full Stack

Trevin Nisbett - Project Manager, Quality Assurance

Professor: Shiva Sharma - IS436 Structured Systems Analysis and Design

Date: November 19, 2023

Deliverable Name: Project Deliverable 4 - Data Modeling and Starting Design

# Data Modeling and Starting Design

[**Data Modeling and Starting Design 2**](#_czcemmawo31v)

[**Online Payroll ERD 3**](#_5pt0hjkiifpo)

[Business Rules 3](#_qsvyhvpw55wm)

[Main Features 3](#_vpnhfaai5o7v)

[ER Diagram 4](#_uhhsbam3i97s)

[Tables 5](#_u8zmpdakr8zo)

[Cardinality and Relationship Description 6](#_5oebkypunyq)

[Normalization 7](#_2u6aacyyn49v)

[**Architecture Matrix 9**](#_23crlsf1qe58)

[**Cloud Computing Architecture Diagram 11**](#_iq0bh9hj8bdn)

[**Hardware/Software Specifications 12**](#_lda5fz7vwu)

[**System Architecture Diagram 13**](#_g2z5722hvvd)

[Diagram Explanation 13](#_s295e1tr5686)

[User Interface Layer 13](#_1mcvdmo30ep)

[Server-Side 13](#_qdxuj5w8shl8)

[Authentication 14](#_7ndxhglgoqvf)

[Backend 14](#_jiqj2lrsnk86)

[End-User Differentiation of Website 14](#_naaf76mlwljt)

[Data Flow and Processing 14](#_spje7vb7w74e)

[Security and Data Management 14](#_36h3ng6dgpjn)

[**Single Alternative Matrix 15**](#_5hi44kkq3tx9)

[Justification 15](#_1jtrgtrugf4k)

[**GitHub Update 17**](#_a8xr1d8roobh)

[GitHub Project Kanban Link: 17](#_ukp5esna7o29)

[**Team Member Information 17**](#_jy4g7b8yt70f)

[Trevin Nisbett - Project Manager, Quality Assurance 17](#_x8qz9ttelva3)

[Dhulfiqar Hussein - Lead Developer I, Full Stack 18](#_h9su21n0xaga)

[Harshitha Sayini - Developer II, Full Stack 18](#_2mhifojcyxei)

[Nishan Subba - Database Administrator 18](#_3eonua9pcal7)

[Aman Gurung - Data/Business Analyst I 18](#_pwbz18obgv7l)

[Chiagoziem Igwe - Data/Business Analyst II 19](#_qgq4s2dz0b8y)

[**Meeting Time 19**](#_qmdtd6i7fg4t)

[Contact US 19](#_dj9z8ozeha77)

# Online Payroll ERD

## Business Rules

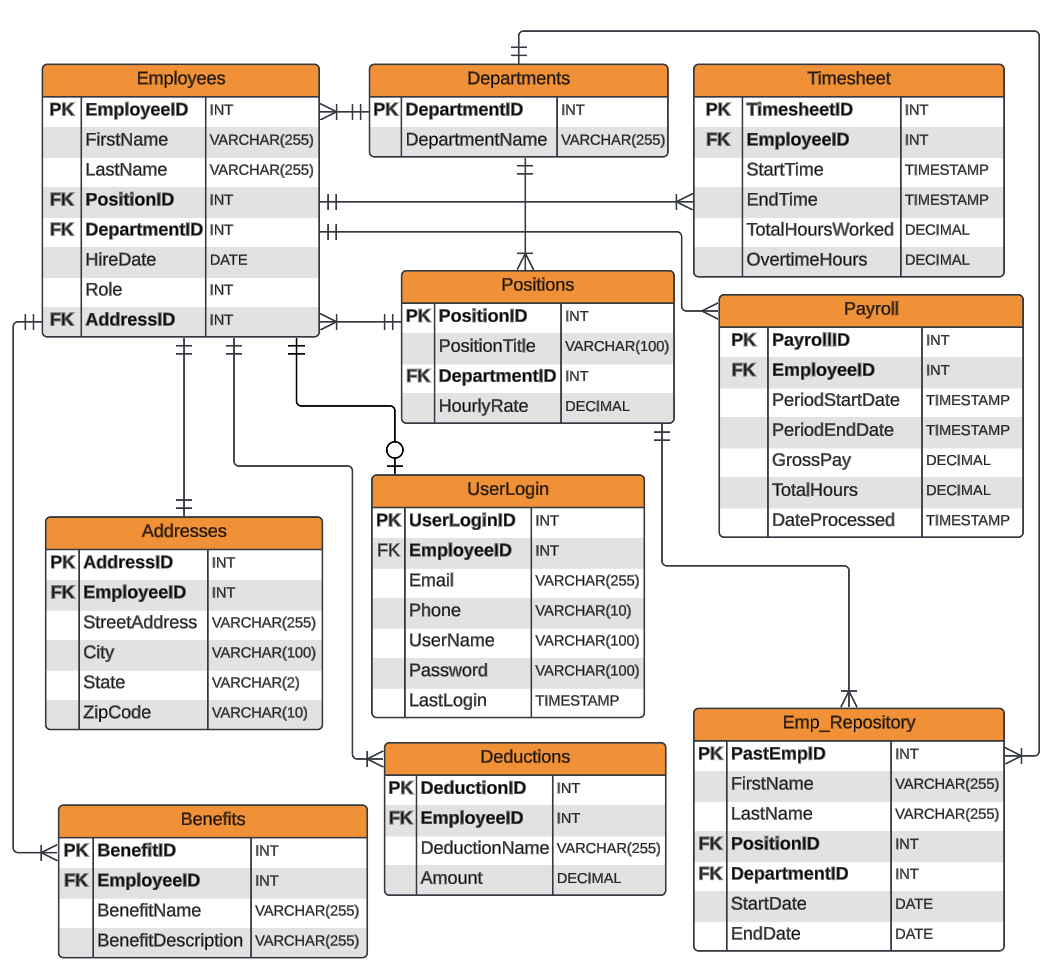
1. Each employee must be associated with exactly one department.
2. Each department can have one or many employees.
3. Each employee must hold exactly one position.
4. Each employee must have exactly one address.
5. Employees can have multiple timesheet entries.
6. Each timesheet entry must be associated with exactly one employee.
7. Timesheet entries include start and end times to calculate the number of hours worked.
8. Each payroll record is associated with exactly one employee.
9. Employees can enroll in multiple benefits.
10. Each employee can have multiple deductions.
11. Employees who leave the company are moved to the Employee Repository table.
12. Each employee has a unique user account for system access.

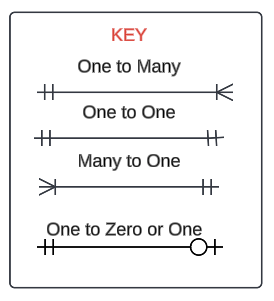
## Main Features

1. Allow employees to enter data and store those data in the database
2. Authenticate employee's information when they log in
3. Validate and Check that the employee is not already registered
4. Add employees to the database
5. Remove employees from the data
6. Generate payroll report by calculating total hours worked and subtracting any deductions
7. Validate overtime requests and update overtime hours in the database
8. If an employee is removed, add a record of it in the repository table for future records

Most of these features will be stored procedures in the database. These procedures will interact with front-end php.

## ER Diagram





## Tables

1. Employees - stores employees' information such as their name, role, hire date, positions, and departments. The Role attribute is about access to the information. 1 = regular access and 0 = Admin access.
2. UserLogin - It also stores their authentication information like email, password, and username.
3. Departments - stores department name
4. Positions - stores positions and their hourly rate
5. Address - stores each employee's address
6. Timesheet - stores how long employees worked on that day and when they clock in and clock out. It will also store overtime hours if they have it.
7. Payroll - table to generate payroll. Stores their pay period, how much they made, and when this report was generated.
8. Deductions - stores deductions name and how much is deducted from gross pay.
9. Benefits - stores benefit name and description
10. Emp\_Repository - a table for former employees. When an employee is removed, their information is stored in this table for future reference.

## Cardinality and Relationship Description

Employees and Department

Cardinality: Many to One (M:1)

Description: Each Employee belongs to one Department, while each Department can have many Employees. The DepartmentID in the Employees table is a foreign key that links each employee to a specific department.

Employees and UserLogin

Cardinality: One to Zero or One

Description: Each Employee may have zero or one User\_Login record, but not more. Whether a UserLogin record exists depends on whether the employee has created their login credentials. The EmployeeID in the Employees table is a foreign key that links each employee to specific login credentials.

Employees and Positions

Cardinality: Many to One (M:1)

Description: Each Employee holds one Position, while each Position can be held by many Employees. The PositionID in the Employees table is a foreign key that links each employee to their position.

Employees and Addresses

Cardinality: One to One (1:1)

Description: Each Employee has one Address, and each Address is associated with one Employee. The AddressID in the Employees table is a foreign key that links each employee to an address.

Employees and Timesheet

Cardinality: One to Many (1:M)

Description: Each Employee can have multiple Timesheet entries, but each Timesheet entry is associated with only one Employee. The EmployeeID in the Timesheet table is a foreign key that links the Employees table.

Employees and Payroll

Cardinality: One to Many (1:M)

Description: Each Employee can have multiple Payroll records, but each Payroll record is associated with only one Employee. The EmployeeID in the Payroll table is a foreign key that links each payroll record to an employee.

Employees and Benefits

Cardinality: One to Many (1:M)

Description: Each Employee can enroll in multiple Benefits, and each Benefit record is linked to one Employee. The EmployeeID in the Benefits table is a foreign key that links each benefit record to an employee.

Employees and Deductions

Cardinality: One to Many (1:M)

Description: Each Employee can have multiple Deduction records, with each Deduction record associated with one Employee. The EmployeeID in the Deductions table is a foreign key that links each deduction record to a specific employee.

Positions and Department

Cardinality: Many to One (M:1)

Description: Each Position is part of one Department, while a Department can have multiple Positions. The DepartmentID in the Positions table is a foreign key that links each position and its department.

Emp\_Repository and Department

Cardinality: Many to One (M:1)

Description: Each record in the Emp\_Repository is associated with one Department, and each Department can be linked to many Former Employees. The DepartmentID in the Emp\_Repository table is a foreign key that links each former employee record to a department.

Emp\_Repository and Positions

Cardinality: Many to One (M:1)

Description: Each Emp\_Repository employee held one Position, and each Position can be associated with many former employees. The PositionID in the Emp\_Repository table is a foreign key that links former employee records to a position.

## Normalization

First Normal Form (1NF)

* Atomic Values: Each column in every table contains atomic values. For example, in the Employees table, attributes like FirstName, and LastName hold single, indivisible values.
* Each column in a table represents a unique attribute, ensuring no redundancy of data.
* Primary Key: Every table has a primary key that uniquely identifies its records. For instance, EmployeeID in the Employees table and DepartmentID in the Department table.

Second Normal Form (2NF)

* All tables in the database meet the 1NF criteria.
* Data that could potentially be redundant is moved to separate tables to avoid repetition. For example, address details are stored in the Addresses table instead of being repeated in the Employees table for each employee.
* Foreign keys are used to link relationships between tables to avoid data redundancy. For example, EmployeeID in the Timesheet table links each timesheet entry to an employee in the Employees table.

Third Normal Form (3NF)

* All tables conform to the rules of both 1NF and 2NF.
* Not fully Dependencies: There are no attributes in any table that aren't fully dependent on the primary key. For example, details specific to positions, such as HourlyRate, are stored in the Positions table rather than in the Employees table.

Fourth Normal Form (4NF)

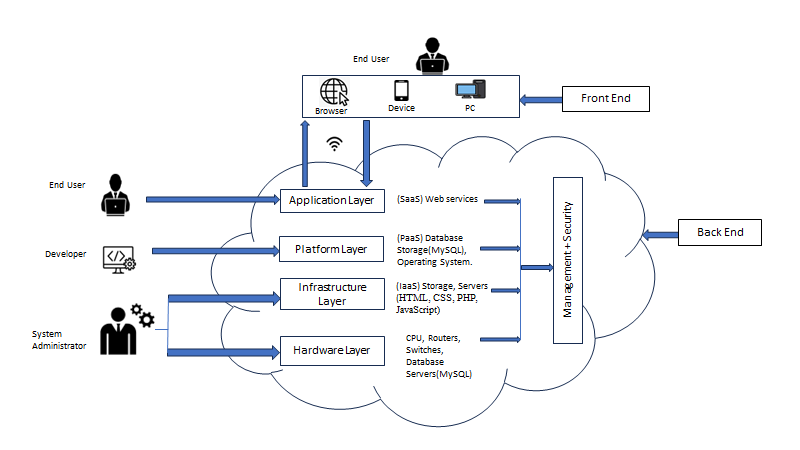
* No Multivalued Dependencies: I designed my database to avoid any multi-valued dependencies.

# Architecture Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirements** | **Server-Based** | **Client-Based** | **Thin Client-Server** | **Thick Client-Server** | **Cloud Computing** |
| **Operational Requirements** | | | | | |
| Portability Requirements |  | ✔ | ✔ |  | ✔ |
| System Integration Requirements | ✔ |  | ✔ | ✔ | ✔ |
| Maintainability Requirements | ✔ |  | ✔ |  | ✔ |
| Technical Requirements | ✔ |  | ✔ | ✔ | ✔ |
| **Performance Requirements** | | | | | |
| Speed Requirements |  | ✔ | ✔ | ✔ | ✔ |
| Scalability/Capacity Requirements |  |  | ✔ | ✔ | ✔ |
| Availability/Reliability Requirements | ✔ | ✔ | ✔ | ✔ | ✔ |
| **Security Requirements** | | | | | |
| Access Control Requirement | ✔ | ✔ |  | ✔ | ✔ |
| Data Encryption and Authentication Requirements |  | ✔ | ✔ | ✔ | ✔ |
| Virus Control Requirements | ✔ |  | ✔ |  | ✔ |
| **Cultural/Political Requirements** | | | | | |
| Legal Requirements | ✔ | ✔ | ✔ | ✔ | ✔ |

In this architecture diagram for our digital payroll system, we want to include a user interface, application server, database server, and cloud storage. The user interface would be the front end where employees would be able to interact with the system. The application server is where the payroll application exists and processes requests from the user interface. The database server would store all the data related to the payroll, such as employees, departments, positions, addresses, etc. Cloud storage would be used to store backup data. The architecture for our payroll system will be designed using a cloud computing model. Based on the matrix, cloud computing is the best choice. This model would allow for cost-effectiveness, scalability, and ease of access. A trade-off of using cloud computing is that it heavily relies on internet connectivity; however, the benefits outweigh the disadvantages, thus making it the best option.

# Cloud Computing Architecture Diagram

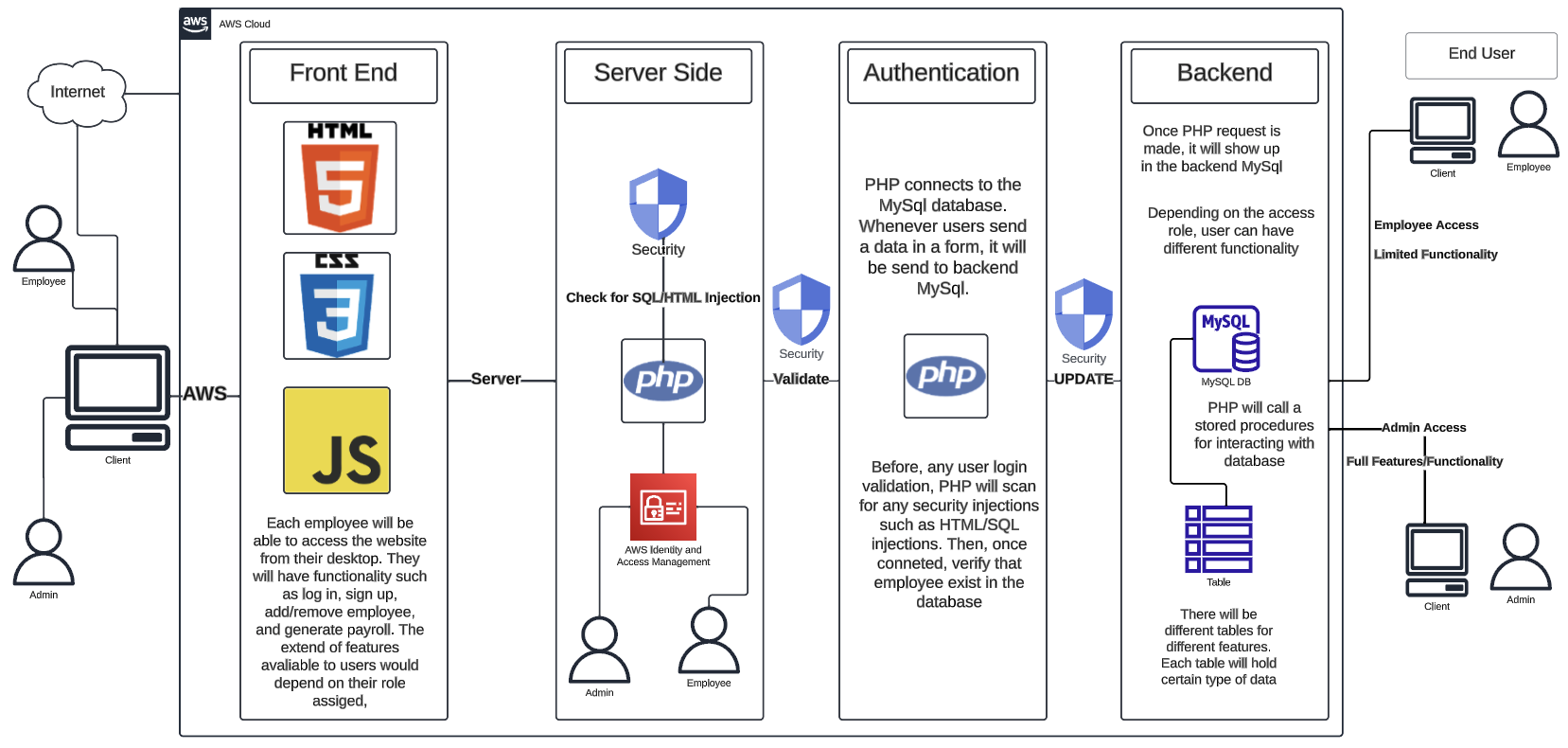


# Hardware/Software Specifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Standard Client | Standard Web Server | Standard Application Server | Standard Database Server |
| Operating System | Windows 10 or higher | Windows 10 or higher | Windows 10 or higher | Windows 10 or higher |
| Special Software | Google Chrome, Firefox | Visual Studio Code | HTML, CSS, PHP, Javascript | MySQL |
| Hardware | PC or Mobile device with internet | Dell PowerEdge T630 Server | RAM, Motherboard | Microsoft SQL Server 2022 Standard |
| Network | LAN, WAN | LAN, WAN | LAN, WAN | LAN, WAN |

These configurations would allow for easy access to the payroll system from anywhere, reduce the need for high-end hardware, and ensure that the data is securely stored and backed up in the cloud.

# System Architecture Diagram



## Diagram Explanation

The system architecture diagram is designed within the AWS cloud environment. This system is structured to ensure security, efficient data processing, and a seamless user experience for both employees and admins.

### User Interface Layer

**Employee/Admin Access:** Users interact with the system through a web-based interface accessible via desktop. The interface is built using HTML, CSS, and JavaScript, making it a responsive and dynamic experience.

**Client Side:** There are various web pages and forms that enable interaction with the server-side components and the database. For example, there is a form to add a new employee, remove an employee, or even generate a payroll report.

### Server-Side

**Security Measures:** All interactions between the client and server are protected by AWS's security features, including checks against SQL/HTML injection to prevent common web vulnerabilities. We will implement a code to prevent any injection-type attacks.

**PHP Processing:** We will use PHP for server-side data handling and database connection. It handles requests from the client, processes logic, and communicates with the database.

Access Management: We will use access management which is built in the PHP file that will dedicate what type of access you have. You can have either the admin or regular employee access. The functionality and features will depend on your access type. When an employee logs in, PHP will send a query to the database and check what type of access is given to that employee. This is to ensure that not everyone has access to unnecessary features and data security.

### Authentication

**User Validation:** PHP scripts handle authentication, ensuring that user credentials are verified against the MySQL database before granting access. It will check for security injections to protect sensitive data.

### Backend

**MySQL Database:** The system uses AWS RDS for MySQL to store and manage all data related to payroll operations. PHP interacts with MySQL through secure queries and stored procedures to perform data manipulation tasks.

**Role-Based Functionality:** Depending on their access level, users can view their payrolls, clock in/out, or if they have admin access, perform administrative tasks such as adding or removing employees and generating payroll reports.

### End-User Differentiation of Website

**Employee Access:** Regular employees have limited access, confined to personal payroll information and time-tracking functionalities.

**Admin Access:** Admin users have full features/functionality, including user management and report generation capabilities.

### Data Flow and Processing

Data flows from the client to the server where PHP scripts process the input. It is then sent to the database for storage or retrieval. The processed data is returned to the server and then to the client for display.

### Security and Data Management

There are several layers of security to safeguard data. This includes front-end validation, server-side checks for injections, secure transmission of data, and role-based access control.

# Single Alternative Matrix

|  |  |
| --- | --- |
| **Criteria** | **Custom Development** |
| Expertise & Customization | High |
| Alignment with Specific Needs | Excellent |
| Adaptability for Future Changes | High |
| Control & Ownership | High |
| Implementation Time | Moderate |
| Cost | Moderate |
| User Adoption & Training | Moderate |
| Integration with Existing Systems | High |
| Scalability | High |
| Technical Support & Maintenance | High |

## Justification

**Expertise & Customization:** The maximum degree of customization and skill is provided by bespoke development. Gorilla Distribution Center's particular needs can be catered for in order to provide a perfect fit for the system.

**Alignment with Specific Needs:** The unique operational subtleties of the center might not be properly aligned with off-the-shelf alternatives. Making each feature unique is ensured via custom development.

**Adaptability for future changes:** The ability to adjust to future modifications in the organization's procedures, structure, or legal needs is offered by a custom solution. Long-term sustainability depends on this adaptability.

**Control and ownership:** Gorilla Distribution Center maintains total ownership and control over the system through custom development. Ensuring that the system evolves in accordance with the strategic objectives of the center is crucial.

**Implementation time:** The customized character of the solution, which directly addresses the center's unique demands, justifies the time investment in creation, even though it could take a little longer to execute than an off-the-shelf solution.

**Cost:** In light of the prospective savings and long-term benefits, the cost of custom development seems reasonable. The return on investment (ROI) is anticipated to be substantial over time, notwithstanding the initial outlay.

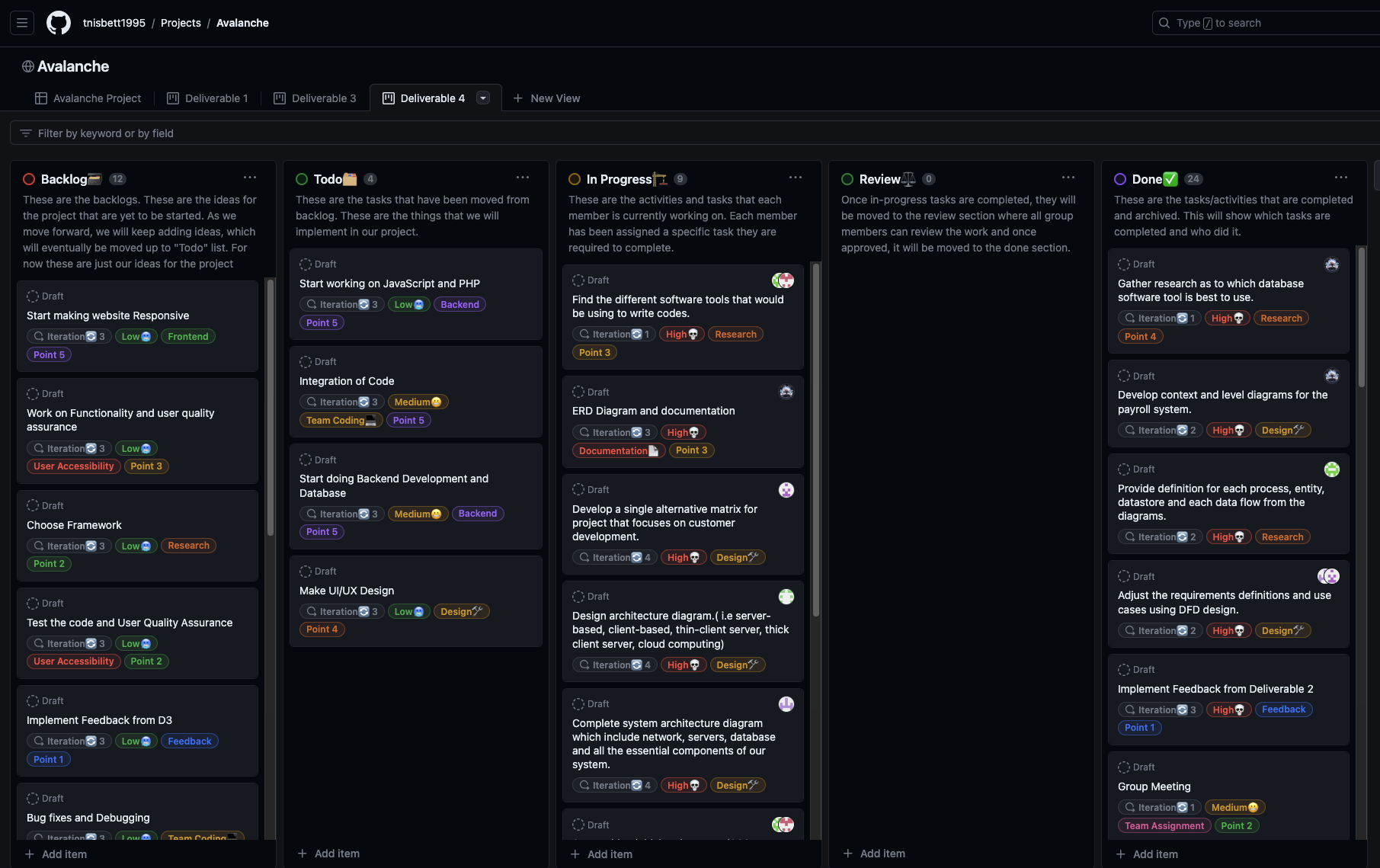
**User Adoption and Training:** The implementation of customized development enables tailored training initiatives, guaranteeing a seamless and comprehensive shift from manual procedures to the digital payroll system for all user segments.

**Integration with Existing System:** At Gorilla Distribution Center, custom development enables seamless integration with any current workflows or systems, preventing interruption and guaranteeing a coherent digital environment.

**Scalability:** Scalability is an important consideration, particularly in light of the center's projected growth. The digital payroll system can grow with the company's expanding workforce and changing requirements thanks to custom development.

**Technical Support and Maintenance:** Dedicated technical support and maintenance are provided by custom development, guaranteeing that the system stays current, safe, and effective. The continued viability of the digital payroll system depends on this support.

# GitHub Update



## GitHub Project Kanban Link:

<https://github.com/users/tnisbett1995/projects/3/views/10>

# Team Member Information

## Trevin Nisbett - Project Manager, Quality Assurance

I am Trevin Nisbett, and I will be the Project Manager for this project. I have experience in Java, SQL, and Excel. I am excited to work on this project and apply my technical skills to the Avalanche Innovation team to have a successful digital payroll system.

For questions or inquiries, please contact me at

Email: [tnisbet1@umbc.edu](mailto:tnisbet1@umbc.edu)

Github: tnisbett1995

## Dhulfiqar Hussein - Lead Developer I, Full Stack

Hello, My name is Dhulfiqar Hussein and I will serve as a full-stack lead developer alongside Harshitha. I have experience in the following areas, SQL, HTML, CSS, JAVASCRIPT, Python, C++, and PHP. I am excited to collaborate with the team and share my technical skills to drive the success of our digital payroll system project.

For questions or inquiries, please contact me at

Email: [dhussei1@umbc.edu](mailto:dhussei1@umbc.edu)

Github: DhulfiqarH

## Harshitha Sayini - Developer II, Full Stack

I am Harshitha Sayini and I will serve as a full-stack developer alongside Dhulfiqar. I have experience working with SQL, HTML, Java, CSS Javascript, and Excel. I will be using my programming skills to aid in the development of our web application for Avalanche Innovation. As a team, we will be committed to creating an application that will be accessible to all users.

For questions or inquiries, please contact me at

Email: [hsayini1@umbc.edu](mailto:hsayini1@umbc.edu)

Github: hsayini7078

## Nishan Subba - Database Administrator

I'm Nishan, and I serve as the Database Administrator for our project. With a strong background in data management and experience in SQL, Python, and Excel, I've had the privilege of working with various databases, including Oracle and MySQL. My passion lies in extracting stories from data and transforming raw information into actionable insights. I'm committed to bringing my data expertise to the Avalanche Innovation team to ensure the success of our digital payroll system project.

For questions or inquiries, please contact me at

Email: [nsubba1@umbc.edu](mailto:nsubba1@umbc.edu)

Github: nissubba1

## Aman Gurung - Data/Business Analyst I

My name is Aman and I will serve as a Data/Business Analyst I for this project. I have experience in SQL, Statistics in Business and Econ, Excel, and Python. I am skilled in producing insightful data visualizations and reports that provide our company with the information they need to make wise decisions.

For questions or inquiries, please contact me at

Email: [amang2@umbc.edu](mailto:amang2@umbc.edu)

Github: Aman440501

## Chiagoziem Igwe - Data/Business Analyst II

My name is Chiagoziem Igwe, and I will serve as a Data and Business Analyst II for this project.

For questions or inquiries, please contact me at

Email: [chiagoi1@umbc.edu](mailto:chiagoi1@umbc.edu)

Github: chiaigwe

# Meeting Time

Weekly Meeting:

Every Sunday @ 8 PM

## Contact US

For any inquiries or additional information related to this proposal, please get in touch with our key project members:

Trevin Nisbett

*Project Manager, Quality Assurance*

Email: [tnisbet1@umbc.edu](mailto:tnisbet1@umbc.edu)

As the Project Manager, Trevin oversees the entire project, ensuring milestones are met, quality is maintained, and all team members collaborate effectively. For any project/proposal inquiry, Trevin is your go-to contact.

Aman Gurung

*Data/Business Analyst I*

Email: [amang2@umbc.edu](mailto:amang2@umbc.edu)

Aman, our Data/Business Analyst I, dives deep into the specifics of the payroll system, ensuring that the proposal aligns perfectly with the business requirements of the Gorilla Distribution Center. For details on the project’s analytical aspects, Aman is your go-to contact.