

# web application Project-1.docx

 Assignment

 Class

 University

---

## Document Details

Submission ID

trn:oid:::1:3020204327

Submission Date

Sep 25, 2024, 3:17 PM UTC

Download Date

Sep 25, 2024, 3:18 PM UTC

File Name

web\_application\_Project-1.docx

File Size

22.8 KB

13 Pages

4,029 Words

21,910 Characters



# 0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

**Caution: Review required.**

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

## Detection Groups

- 
**1 AI-generated only 0%**  
 Likely AI-generated text from a large-language model.
- 
**2 AI-generated text that was AI-paraphrased 0%**  
 Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

### Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

## Frequently Asked Questions

### How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (\*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

### What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.



## Online Grocery Store Project

### Team Introduction

This project is being executed by a team of junior software engineers, and our proposed application is the Web-based eCommerce Online Grocery Store. The idea behind every project is to deliver a rich user experience of customers creating accounts, viewing grocery products and making transactions. Individual roles and responsibilities were then delegated to every team member with focus to the skills they offered and the development phases.

### Idea Pitching

The concept behind the Online Grocery Store project is that eCommerce technologies shall be utilized in creating a user friendly, time efficient, secure platform for online purchase of groceries. Since e-commerce has gained popularity and has been playing a vital role in our people's lives, particularly in urban areas, our team aligned our efforts toward minimizing the inconvenience that people experience while grocery shopping. Groceries can be scrolled for, the specific product can be viewed and bought, new items can be added to a cart, and a purchase can be made all within protected environment in a conveniently designed interface.

This platform's main purpose is the improvement of the shopping process as a user entertainment source, and not only as a functional tool; features such as the real time inventory availability which makes users view only the items that are currently in stock are also typically implemented here. Besides, the system will propose related products that the customer has used before, which bring convenient service to the user, making shopping easier and quicker. The quick checkout process follows the techniques that reduce a number of steps as much as possible and increases the level of protection, proposing different ways to pay for the selected products or services and simple navigation from the step of choosing a product or service to the step of payment.

The system has been designed for requested scalability and flexibility that would enable it accommodate all forms of customers, including the giant grocery wholesalers to the humble small businesses. As we developed the platform, key goals included:

- **Convenience:** Simplifying the shopping process.
- **Personalization:** Offering a tailored shopping experience.
- **Security:** Safeguarding user data and transactions.

- **Scalability:** Accommodating future growth and potential integrations with other platforms.

## Project Planning

Project management planning plays an important role for the proper execution of the Online Grocery Store project and for the project to be completed on time and on scope. Agile approach was used in this project and this entailed that we were able to divide the project into more flexible units of development known as sprints. This made it possible to constantly check and update the status from the various stakeholders, so that improvement could be made each time depending on the feedback from users.

As a result of managing the project, we used project management tools such as Trello to assign and track tasks and continually check if the members of the team understood what was expected of them. Each team member was assigned a specific role based on their expertise:

**Lead Developer:** Mainly involved in the architecture of backend and the microservices, APIs and databases were also my responsibility.

**Frontend Developer:** Specialized in the user interface (UI) aspect of the platform in order to maintain functionality as well as aesthetics across multiple devices.

**Database Manager:** It involved establishing and overseeing the database with various products, user and order data required to be efficiently collected and stored.

**Quality Assurance (QA):** Engaged in testing the application for apparent bugs with the aim of achieving the project requirements and objectives on performance and usability.

We chose Agile because not only it allows us to create change as the project develops, but also to consider any modifications in the functional requirements or the feedback we might receive. Sprint delivered functional increments of the product with daily team meetings and end of sprint review to evaluate and respectively, to set objectives for the next phase of the endeavor.

## Requirements Elicitation

At the beginning of the work, we obtained requirements from the use of user stories to guarantee that the system developed suited the different roles of customers, administrators, and business owners. Every user story described a feature broadly from the point of view of the user which allowed the designer to consider use experience.

Some key user stories that shaped the platform include:

1. *As a user, I want to create an account so that I can track my orders and maintain a purchase history.*
2. *As a user, I want to easily search for grocery items by categories or specific keywords to quickly find what I need.*
3. *As an admin, I want to manage the inventory in real-time, updating product details and availability to reflect current stock.*

In addition to user stories, we created **use cases** that detailed how different components of the system would interact with users. We also developed **prototypes** using tools like **Figma**, providing early visualizations of the user interface. These prototypes helped gather feedback on the design and functionality before full-scale development began.

### Refinement and Specification

Once the initial requirements were identified, we focused on refining them to develop a more detailed set of system specifications. This phase involved careful consideration of key system components:

**Database design:** We made sure that all the data that we collected from the user, order and product were well organized and secured in the relational database models. Concerning the structure, the concern for scalability gave the database room to grow as more users and products joined the system.

**UI/UX improvements:** So, relying on performers' feedback towards first prototypes, we added the refinement of such elements. This involved improving the page navigation, adopting a mobile responsive design, and incorporating low mental burden design features.

**Security measures:** To prevent leakage of important info such as consumer's credit card information and login credentials, we incorporated secure measures. This comprised transmission security through encryption; containing authentication for access to premises and network; and RBAC to limit higher privilege features for management to only approved users.

### Further Refinement

However, progress was made with the aim to further develop the platform throughout the research process using different types of testing parameters and users' comments. Several critical areas that saw improvements included:

**Search efficiency:** The search function was enhanced for faster and better results; we improved the keyword match, category, and suggestions.

**Faster checkout options:** To cut the number of steps at the checkout, any unnecessary steps were removed, and to reduce the time spent when going through the payment gateway, the integration of the payment gateway was optimized.

**Inventory accuracy:** To address this issue, we put measures of real-time inventory that means a user will not be able to view or select a certain product if it is out of stock.

Specific measures that had informed this process included the page load speeds, how the system performed under various loads, and from the test users themselves feedback which pointed at existing constraints in the user experience.

### High-Level Design

The high level design of Online Grocery Store utilized the MVC model which is also called as Model-View-Controller model. This design approach ensured the separation of concerns, making the system more maintainable and scalable:

**Frontend (View):** Developed using HTML, CSS, and other cutting-edge web improvement technologies such as React.js, the frontend provides a pleasing, adaptive homepage. It makes the components to render faster than normal practice, user experience enhances through smooth navigation and fewer page refresh.

**Backend (Controller):** Written in Node.js, the backend is responsible for all calculations and all interactions between the frontend and the database. The service layer gives RESTful APIs to the front end in order that data will stream non-stop between the consumer and the server.

**Database (Model):** The specific technology that was selected to store important, and often times sensitive information, includes a MySQL database for information like user details, product list and transaction log. It also ensures the data is consistent and of high integrity since this relational database model organizes the data.

### Low-Level Design

In low-level details of the Online Grocery Store, the design was done at the lowest level which defined how the system operates and how it will take various interactions. This stage centred on database structures and security as well as the integrity of the entire system.

Database schema: The actual work in building the framework was very deliberate in that we initiated a series of tables that defines the database including tables for users, products, orders and transactions among others. Every table was created with special attention to relations between them and data integrity, so each value was presented in one instance only. For example:

- ✓ Users table: Used for saving back basic user information, for instance, names, email addresses and passwords in encrypted form.
- ✓ Products table: Contained product names, description, categories, prices, and the quantity in stock.
- ✓ Orders table: Stored order information, that associated particular users with products through the use of a foreign key.
- ✓ Transactions table: Generated details of payment mode and payment status of each order and made sure that the sample purchase history was trackable and reportable.

Indexes were set up in advance and constraints were used to make searches or operations, such as looking up products or processing orders fast, as the data base can expand rapidly.

**Security features:** Since the platform is designed for secure processing of user data and transactions, an emphasis on security was made at all the layers of the application. Some key implementations included:

- ✓ Hashed encryption: Passwords were not plain, instead, we hashed them through secure processes such as bcrypt or Argon2. It also made it very hard for adversaries to gain the original plain text password in cases of a data breach.
- ✓ Two-factor authentication (2FA): We also saw it was necessary to integrate two-factor authentication for the users' logins. This meant that users had to enter another form of ID (for example, a code such as one sent to the user's phone or email) in addition to their password. This greatly helped in minimizing the possibility of the code being breached despite someone you knowing the password.
- ✓ Role-based access control (RBAC): We introduced RBAC to secure strictly limited administrative processes. One could only access some of the most sensitive actions like inventory control, placing, and receiving orders, or altering user accounts if one

had the proper roles assigned to their user accounts (like being an administrator, for example). These measures ensured that the part was not taken out deliberately by someone working for the organization or an accident occurred whereby wrong parts were fitted on the system. Also, the admin dashboard and all related API endpoints were enhanced with strong access controls with logs to track activities within the system.

As a result of the low-level design work carried out in detail at this stage, we were able to guarantee that the final system which was being implemented, would be not only secure but also able to effectively process and protect user and business data. This also formed the basis of scalability in that the platform would be able to embrace more traffic and transaction as it empowered the end users.

### Programming Phase

During the programming phase, directly adhering to the code standards was followed to achieve the maximum efficiency, code reusability factor and coding using Team collaboration. One of the key objectives was to make development iterative and maintainable in order to quickly make changes and deliver good results.

- **Code reviews:** That, all features which were designed passed through peer reviews from colleagues and experts in the field. These reviews were done by peers who were developers, in order to ascertain that the code that we were committing to our project, adhered to the highest standards of quality, functionality, and security. Code reviews enabled us identify possible bugs before integration, improve standardization across the code and enhance the team training. It also made the review process more collective where every member was conversant with the general layout and working of the application fostering unity in the development process.
- **Modular coding:** We created clean code by following all principles of code such that each module contained an individual responsibility only. This modularity paid off in making the code reusable, especially between different parts of the app and in making it easier to correct bugs or add features without interacting with other parts of the system. Thereby we minimized development of the so called technical debt – the future costs concerned following and improving the system.
- **Version control:** To achieve co-ordination and to manage the changes we followed the Git repository for version controls. It enabled one to carry out different parts of the



project at the same time without coming across the other's work. Version control presented in this paper helped to reveal the history of changes in code for a better and simpler identification of a problem. However, using the branching strategies, we were able to correctly split features under development and look for a way of integrating them back into the master code when they're stable and reviewed.

### Algorithm Implementation

As part of enhancing the functionality and user experience of the Online Grocery Store, we implemented key algorithms tailored to improve search efficiency and personalization:

- **Product recommendation algorithm:** To enhance the user experience we built a recommender system that was capable of recommending products similar to others that the users had purchased in the past. This was developed based on a simple algorithm that relies on patterns and trends of purchasing behavior to suggest other related items users might be interested in hence creating a more customer-oriented marketplace. In the course of several iterations, the algorithm can be refined and improved by the integration of machine learning models.
- **Search optimization:** To improve the user experience we implemented index-based search for our site. This made it possible to access search results in few seconds any time information was requested since the size of the product database would inevitably increase with time. With these optimizations, users could not only locate specific grocery items via keywords or categories of products, but also specific details of the product. The given search functionality was built with search capability ability to expand with the increasing product catalogue, all the time remaining fast.

### Security Implementation

One of the concerns during the developmental stage was security since the site collects personal information and payment information from the users. We incorporated a variety of security measures to safeguard the system against potential threats:

- **HTTPS:** All web components of the application were deployed over the https protocol which embeds all the data sent between the client and the server. This measure ensured that users' passwords, account details and any other form of information never reached the wrong hands.

- **SQL injection prevention:** This is to avoid cases where some users try to manipulate some parameters in the input fields in what we know as SQL injection attack, we used prepared statements in all the queries to the database. Prepared statements made it possible to only accept MySQL commands that were safe when executed, so user input was cleansed prior to being executed as part of SQL commands, thereby protecting the database from these particular hostile attacks.
- **Data validation and sanitization:** For anti-XSS and other forms of input-based attack, we employed stringent methods of data validation. Each form and each URL parameter passed from users to the application was checked and sanitized. This made it impossible for the system to be manipulated by wrong scripts or wrong entries from the users. Further, client side and server side validation techniques were employed in order to maintain the quality of the input.

### Ethical Considerations

Ethical guidelines were followed throughout the project, focusing on:

- **Data privacy:** Only acquisition of relevant data was made, and every information regarding the users was encrypted.
- **Inclusive design:** We made it possible for everyone including persons with disabilities to this platform by following web standards.

### Testing Phase

The testing phase was particularly important to fulfill the quality and performance requirements for an eCommerce application. As a result, we used different approaches in testing to ensure provision of a solution that would capture every aspect of functionality, usability, and security. This extensive testing also helped to reduce the probability of defects, and as far as defects were concerned, the system functioned optimally as predicted in different scenarios.

- **Unit testing:** All the modules of the application, including the login sub-system, search and the checkout system were tested using functional unit tests. These tests proved that each function or module should do its operations correctly if no other functions or modules are called. This early detection of bugs eliminated likely hood of large problems during later stages of development.

- **Integration testing:** The two tests that were carried out after unit testing were integration tests, which aimed at testing how well various sub-systems of the application interfaced with each other. This included checking if any or all of the front-end interfaces implemented submits and interacts with the back-end server as anticipated and checking if the data stored in the database was stored and retrieved as required. These tests were important to ensure that the overall process of data operation from the user point of view to the back end point of view was free from all sorts of error.
- **User testing:** However, user testing was also done for obtaining the real-life experience feedback on the usability and performance of the platform and overall experience of the user. Several scenarios were assigned to users including signing up, searching for products, placing the products into the cart, and completing the purchase. With their feedback, problems in the user experience, for instance, where the navigation was somewhat confusing, or when loading time was unreasonably long, were noticed and the user interface was optimized.

### Quality Assurance

To maintain high standards of code quality, we applied several quality assurance metrics during the development and testing phases:

- **Code coverage:** These are the following technique: code coverage, which checked that the automated test coverage at least 90% of the code. This metric made me feel that most part of the application functions had been tested, thereby minimizing a situation where untested code can head to production environment and cause havoc. High coverage also enabled us to modify the system code without fearing to introduce various unlooked for errors.
- **Defect analysis:** Any problems or defects as observed during testing were well documented including the corresponding category. In this case, utilizing the systematic defect analysis process, all bugs were analyzed and the most basic causes of the defects were determined and eliminated. This made it possible to strengthen the system so that in case of deliberate attacks, the same defects that were identified could not easily be repeated again after the system has been deployed. That we eliminated all the existing problems before its release made it have a higher quality and stability of the application.

## Deployment Strategies

When it was time to put the system out to production, a CI/CD pipeline was set up to build the system and deploy new features, while maintaining the existing functionality.

- **Jenkins was our principal application for handling builds, tests, as well as deployment. With the help of this tool it was rather simple to incorporate a fresh code for the application, since after the changes all unit and integration tests are run to check for any mistakes before deploying the application. Every feature or bug fix first rolled out to a separate staging environment, so at least two separate tests could be performed to make sure there were no problems on the live platform. After tests passed at this site, the changes were transferred to the live site at the same environment.**

## Maintenance and Future Trends

Following deployment, we established a clear strategy for post-launch maintenance to ensure that the system remained functional and responsive to user needs:

- **Bug tracking:** For bugs and system issues that cropped up after the launch we used Jira. Jira enabled us to manage bugs at a glance, prioritize them, and determine actions for their resolution so that any problems users encountered were soon addressed without infringing on their experience. In this way, we were able to organize these bugs and recognize which were the most important to fix before focusing on other areas of improvement.
- **System monitoring:** In order to increase the level of system dependability we put into use software that would systematically control variables such as server availability, database utilization and frequency of errors. Such tools enabled us to predict problems – for example, if a program is too slow or if a system is open to an attack – before they posed risks to users. Monitoring also serve to organize its growth, to guarantee the system could accommodate increasing amount of users in the future.

Looking ahead, we identified several future trends and innovations to enhance the platform further:

- **AI-driven recommendations:** For better evaluation of product recommendations, we are also considering the use of artificial intelligence and machine learning to provide

customer-tailored recommendation. Whereas most current suggestions use simple purchase histories, these gradations of algorithms would employ search histories, clicks, and other real-time behavioral patterns.

- **Blockchain for payments:** We are also in a process of implementing the payment system using the blockchain technology tool to increase digital payments' security and transparency. Users can experience faster and even safer payments and may trust the platform more due to new unique characteristics such as decentralized transaction histories available in the use of blockchain.

Such an approach helped to cover not only the given brief but also to create a flexible and secure foundation for the future development of the Online Grocery Store.

## How to Run the Project

Read through the guide below to understand how to set up the project, the requirements needed for the Online Grocery Store.

### 1. System Configuration

- **Install Node.js:** Before going to the next step, make sure Node.js is installed on your machine. Download and install it from the official website: Node.js.
- **Install MongoDB 3.4 Stable Version:** MongoDB will be used to perform database management duties. You can download and install v3.4 from MongoDB.
- **Install a Code Editor:** We encourage you to leverage on Visual Studio Code, Sublime Text or any code editor of your preference. Get the latest build of Visual Studio Code from [here](#).

### 2. Install NPM Packages

Once the system is set up, install the required NPM packages to ensure that the project runs smoothly:

- The next step leading to implementation involves opening your terminal or command prompt to the project directory folder.
- Run the following command to install all the required NPM packages:
- `bash`

- Copy code
- npm install
- This command will in turn install all the dependencies as defined in the package.json the common ones being express, mongoose, and others.

### 3. Import Sample Database

To run the project with sample data:

- From the database folder, call the products collection into your MongoDB database. This helps to initialise samples of data in the system should it be required to display in a Store.
- Open MongoDB.
- Go to the MongoDB database folder and either use the command-based MongoDB tools (mongoimport is one of them) or the graphical tool called MongoDB Compass.

Example command for importing the data using mongoimport:

```
mongoimport --db groceryDB -c products -f./database/products.json --jsonArray
```

Do not forget to replace the path if necessary.

### 4. Start the Express Server

After importing the sample data and installing the required packages, follow these steps to start the application:

Ensure that all necessary Node.js packages are installed by running:

```
npm install
```

Start the Express server on localhost:3000 through the command;

```
npm start
```

Open your browser and navigate to <http://localhost:3000/> By invading the URL [www.OGShop.com](http://localhost:3000/) to the URL [//localhost:3000/](http://localhost:3000/) it can be accessed that the Online Grocery Store.

Summary of Steps

Before all, make sure that Node.js, MongoDB version 3.4 and any code editor is properly installed.

To run this application on localhost, install all the necessary NPM packages by simply typing `npm install` in the terminal of the root folder.

The database folder should now be imported into MongoDB to include the products collection.

Start the Express server by running `npm start` and access the application at `http://localhost:3000/`.

.