

UC Reviews

by

Adam Tulloss, Chase Staggs, Pratik Chaudhari, Elliott Phillips

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

© Copyright 2025 Adam Tulloss, Chase Staggs, Pratik Chaudhari, Elliot Phillips

The author grants to the School of Information Technology permission
to reproduce and distribute copies of this document in whole or in part.



4/14/25

Adam Tulloss



4/14/25

Chase Staggs



4/14/25

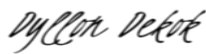
Pratik Chaudhari



4/14/25

Elliott Philips

1/23/25



Dyllon Dekok, Advisor

University of Cincinnati
College of
Education, Criminal Justice, and Human Service

Contents

<i>UC Reviews</i>	<i>1</i>
<i>Contents</i>	<i>4</i>
LIST OF FIGURES	5
ABSTRACT	4
1. INTRODUCTION	1
1.1 Introduction.....	1
1.2 Problem	1
1.3 Solution	1
1.4 Project Goals	2
1.5 Overview.....	2
2. DISCUSSION	4
2.1 Project Concept	4
2.2 Design Objectives	4
2.3 Methodology and Technical Approach	5
2.4 User Profiles/Personas	5
2.5 Use Case Diagrams	7
2.6 Technical Architecture	8
2.7 Testing.....	9
2.8 Budget.....	10
2.9 Project Plan	11
2.10 Problems Encountered and Analysis of Problems Solved.....	13
2.11 Recommendations for Improvement	14
3. CONCLUSION	15
3.1 Lessons Learned.....	15
3.2 Abilities and Skills Developed Throughout Project	15
3.3 User Needs Reflection.....	15
3.4 Security Principles and Practices Reflection	17
3.5 Project Conclusion and Future Work	18
4. REFERENCES AND APPENDIX	19
4.1 References.....	19

LIST OF FIGURES

1. Figure 1. Use Case Diagrams
2. Figure 2. Entity Relationship Diagrams

ABSTRACT

The UC Reviews project aims to develop a responsive web and mobile application allowing students at the University of Cincinnati to leave detailed reviews across various aspects of campus life, including housing, dining, faculty and campus facilities. By providing a platform for in-depth feedback beyond simple ratings, UC Reviews addresses limitations in existing methods, such as the HappyOrNot® kiosks, which lack contextual insights. By implementing features such as liking and disliking reviews, as well as reporting mechanisms, UC Reviews aims to foster a supportive community, encouraging genuine feedback that can guide actionable improvements across the campus.

1. INTRODUCTION

1.1 Introduction

The University of Cincinnati's (UC) student experience is multifaceted and shaped by various aspects of the campus from housing and dining to academic experiences and social interactions. Unfortunately, current feedback mechanisms, such as rating kiosks, do not fully capture the depth and context needed to drive targeted improvements. By allowing students to provide both written and score-based reviews for aspects of campus life, UC Reviews will offer valuable insights for enhancing student satisfaction and campus operations.

1.2 Problem

At the University of Cincinnati, existing feedback systems tend to collect limited data, often focusing only on surface-level satisfaction. For instance, the HappyOrNot® kiosks on campus allow users to rate their experience, but offer no option to leave comments or context (University of Cincinnati, 2023). Without additional context, it becomes challenging to extract actionable insights or identify specific issues. UC Reviews addresses this gap by enabling users to post detailed textual reviews, allowing university administrators to better understand and resolve concerns that affect student life (Kegley, 2024).

1.3 Solution

UC Reviews presents a solution by introducing a comprehensive review platform tailored to the UC student body. Through this platform, students can leave detailed reviews on housing, dining, courses and faculty, providing meaningful context to their experiences. The added context allows

the university to identify specific areas needing improvement and empower campus entities to respond to feedback constructively.

1.4 Project Goals

Our implementation of the UC reviews project was made with the following goals in mind:

- **Comprehensive Feedback Collection:** Allow for students to provide detailed feedback on various aspects of campus life through textual reviews and star ratings.
- **Intuitive User Experience:** Develop an intuitive, responsive interface that allows students to navigate and leave reviews easily.
- **Privacy and Safety:** Implement guidelines and tools to ensure reviews are safe, respectful and informative while protecting student privacy.
- **Collect Actionable Data:** Offer data-driven insights that highlight key areas for improvement around campus, ultimately enhancing the student experience.

1.5 Overview

This paper aims to provide a brief analysis of the feedback mechanisms currently in place at UC, identify their shortcomings, and propose a new solution, UC Reviews, designed to provide a simplified platform for providing feedback about campus experiences. It begins by introducing the challenges faced in gathering actionable feedback and details the objectives of the proposed platform. The discussion section covers the project concept, design objectives, technical architecture and reviewal/testing processes adopted to ensure quality. The conclusion reflects on lessons learned and skills developed while creating this project, as well as plans for future

improvements. By addressing gaps in existing feedback systems, this paper illustrates how UC Reviews could serve as a tool for continuous campus improvement.

2. DISCUSSION

2.1 Project Concept

The original idea for UC Reviews was sparked by UC students expressing the need for a platform to openly share their experiences regarding campus life, particularly with housing. Recognizing that current feedback mechanisms do not allow for detailed or constructive feedback, the project team sought to broaden this initial idea to encompass a wider range of campus elements. UC Reviews now includes a platform for students to review not only housing but also dining, faculty, courses and other campus services.

2.2 Design Objectives

UC Review's design objectives focus on usability, accessibility, and data integrity. Key objectives include the following:

- **User-Centric Interface:** The interface should be intuitive; it should be easy to navigate and review different aspects of UC.
- **Authentication and Privacy:** The website should ensure only UC students can leave reviews.
- **Review System:** The system should allow users to leave reviews along with star ratings.
- **Moderation and Reporting Tools**
- **Entity-Specific Pages**

2.3 Methodology and Technical Approach

Agile development methodology, organized around two-week sprints, will be used to create UC Reviews. By enabling the team to prioritize tasks, respond quickly to feedback, and modify development goals as the project progresses, this methodology fosters adaptability. Agile also promotes ongoing cooperation between developers and stakeholders, which is particularly helpful in our situation because we frequently take peer, instructor, and user feedback into account. The team has review meetings at the conclusion of each sprint to demonstrate on finished features, get input, and make plans for the following one. Throughout development, this iterative process guarantees that the application stays in line with user needs and provides value gradually.

2.4 User Profiles/Personas

Based on the intended functionality of the app, we developed fictitious user personas to illustrate how UC Reviews can benefit its users. By keeping actual user needs and objectives at the center of development, these personas serve as representations of typical UC Reviews application users and help in guiding design decisions. By reflecting distinct user behaviors, frustrations, and motivations, each persona enables us to create a solution that is more user-focused and empathetic. The User Persona is as follows:

Name: Bea R. Catlin

Age: 20

Year: Sophomore

Major: Psychology

Background: Bea is an out-of-state student adjusting to live at the University of Cincinnati. She lives in campus housing and frequently uses the dining facilities, attends various social events

and has a full course load. Being detail-oriented, she values transparency and enjoys sharing experiences with others to help them make informed decisions.

Demographics:

- **Location:** Cincinnati, Ohio
- **Education Level:** Undergraduate student
- **Technology Usage:** Primarily uses a cell phone, but uses a laptop for completing coursework

Goals:

- **Stay Informed:** Bea wants to know what others think about specific campus services and locations before trying them herself
- **Share Experiences:** She wants to provide constructive feedback, especially on housing and dining, to help others.
- **Promote Change:** Bea is passionate about improving the student experience and believes in the power of feedback to prompt change around campus.

Frustrations:

- **Lack of Contextual Reviews:** She feels that existing feedback tools, such as kiosks, don't allow her to explain her experiences and suggest improvements
- **Inability to See Peer Feedback:** Emily wishes that she could see what her peers say about various aspects of campus to better guide her decisions.

2.5 Use Case Diagrams

Use case diagrams are essential tools in system design as they provide a high-level visual representation of how different types of users interact with a system. For UC Reviews, use case diagrams help outline the core functionalities available to students, faculty, and administrators. These diagrams ensure that the user requirements are fully captured and that each system function is aligned with project goals. Below is the use case diagram specifically developed for our web application.

This use case diagram describes the role and interaction of users, administrators, and the system itself while working with the application called UC Reviews. It includes various types of usages of different users for viewing reviews, leaving reviews, rating entities, and interacting with other reviews. It also represents the admin's part in maintaining the content, approving new entities, and responding to users' reviews.

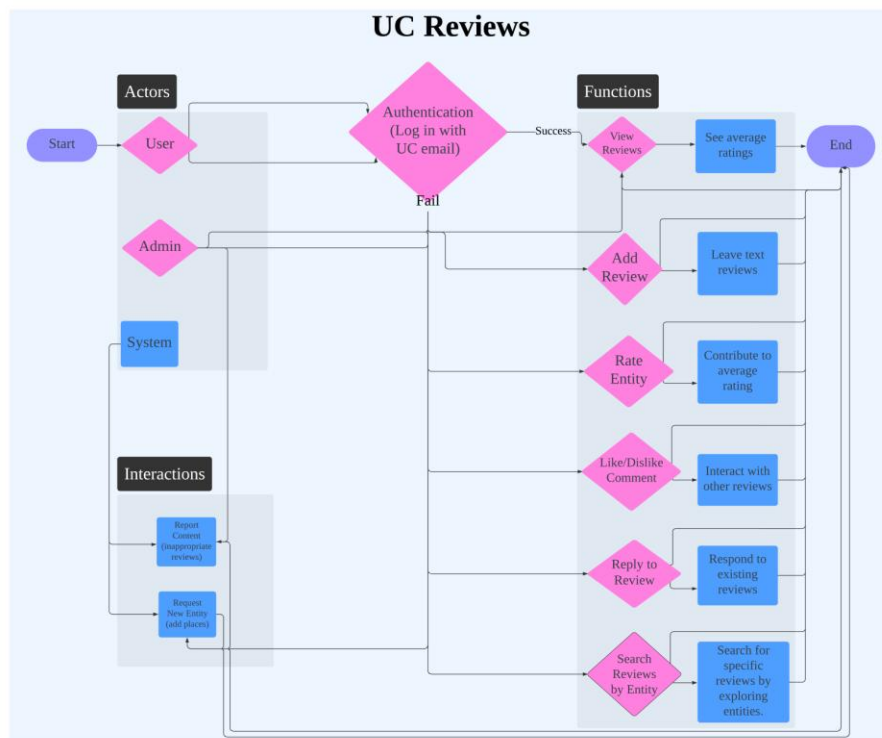


Figure 1 Use Case Diagrams

2.6 Technical Architecture

This application consists of 3 components, the backend which is a RESTful API built using C# and Entity Framework, a frontend built in Angular, and a Database which is Microsoft SQL Server. The backend interacts with the database and receives and responds to requests from the front end. It consists of the following:

- Controllers – Each entity has a dedicated controller that handles the HTTP requests. Controllers delegate business logic to their corresponding service classes.
- Services – Services contain the application logic and work as intermediaries between controllers and repositories.

- Repositories – Repositories abstract database operations, ensuring data access is performed efficiently.
- Mail Service – The mail service handles communication with Gmail. This is currently used to send temporary codes to users, much like a password-less login system.

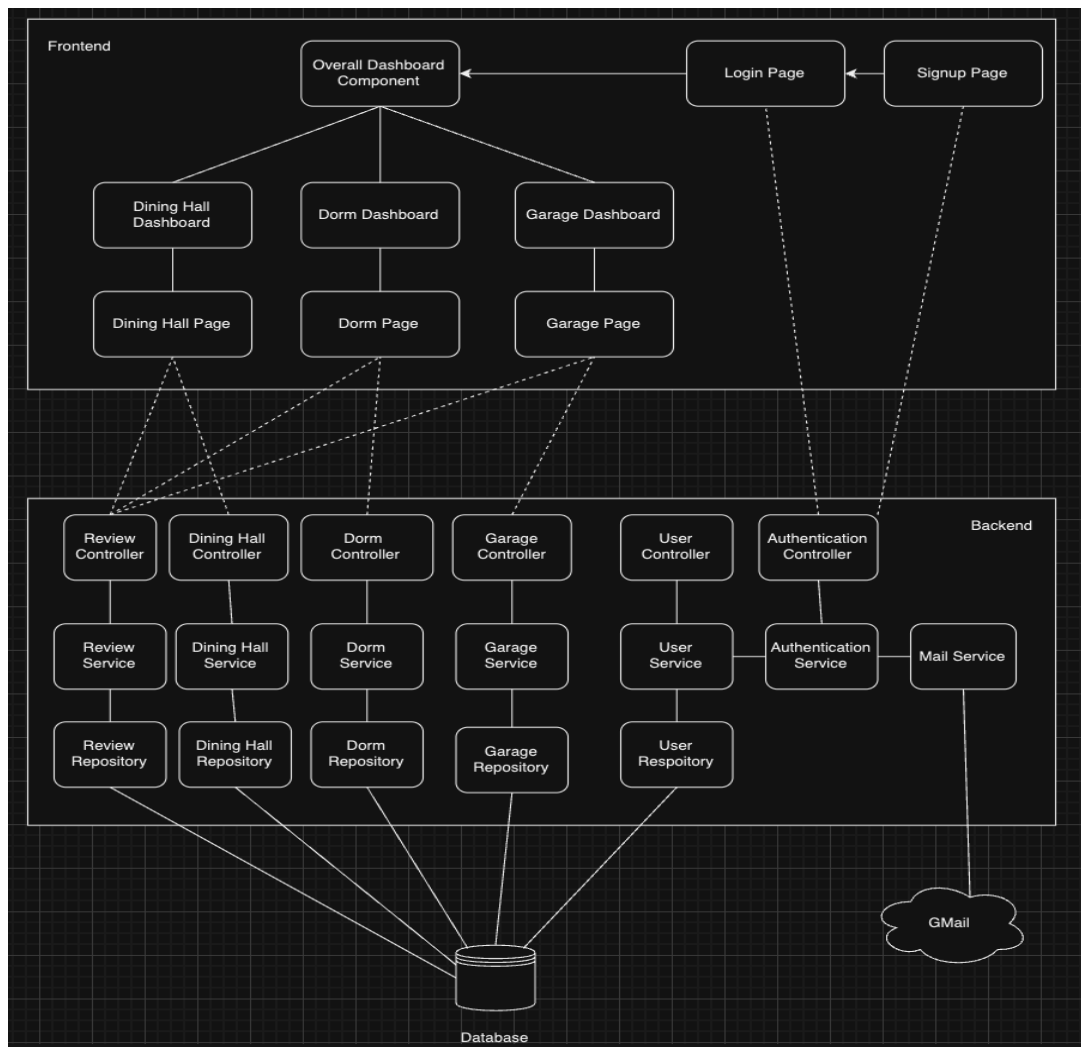


Figure 2 - Entity Relationship Diagram

2.7 Testing

Testing is an essential aspect of software development employed to guarantee that an application behaves as anticipated, contains no critical defects, and provides a uniform user experience. As

part of our UC Reviews initiative, we integrated a Quality Assurance (QA) process aimed at achieving high levels of code quality and functionality. Quality Assurance (QA) is the act of making sure that an application works appropriately, is reliable, and complies with the expected standards prior to release or integration.

In the Quality Assurance process, each feature branch requires a review and one approval before it can be merged into the main branch. The approver pulls the feature branch locally, runs the code, and verifies functionality to ensure that everything works and appears as intended. Unit testing is also incorporated to validate individual components and catch potential issues early. This process helps maintain high code quality and smooth integration of new features.

2.8 Budget

Building UC Reviews took an estimated 150 hours of development time. All the software used to build the application was free and open-source, so there were no associated costs. Creating the project reports required an estimated 25 hours.

The cost per hour for the website's development (frontend and backend) was calculated using the median salary of a software developer and technical writer in Cincinnati, Ohio, which were \$115,643/year and \$84,921/year respectively. These annualized salaries were then turned into hourly wages through the formula:

$$\text{Hourly Wage} = \frac{\text{Annual Salary}}{\text{Hours Per Week} \cdot \text{Weeks Worked Per Year}}$$

Then rounded these up to the nearest hundredth. Overall, the project would cost about \$9360.

NO.	Item	Hours	Price/Hour (\$)	Line Item Total (\$)
Software				
1	VSCode	1	0	-
2	Docker	1	0	-
3	MSSQL	1	0	-

4	Angular	1	0	-
Labor				
5	Website Frontend	70	55.60	3,892.00
6	Website Backend	80	55.60	4,448.00
7	Project Reporting	25	40.83	1020.75
Total Cost			9,360.75	

2.9 Project Plan

The Project Plan defines a range of key features and improvements that will enhance the functionality, usability, and accessibility of the UC Reviews web application. These intended additions draw on the application's existing scope and are intended to meet user needs more effectively, facilitate future expansion, and enhance overall user experience. Each component has been thoughtfully weighed to ensure that it supports the platform's intentions — providing students with a secure, informative, and accessible forum for leaving and viewing feedback regarding campus services and facilities.

Course Review Module

Add a module for courses to be reviewed. This module will be somewhat different from the other modules due to the large number of courses offered by UC, so users will be able to add a course themselves.

Pagination

Adding pagination to the dashboards and review pages would speed up querying pages and prevent overwhelming the user with too much information. For example, while there may be thousands of courses, it wouldn't make sense to show the user all of these at once. It could lag

the website's frontend and backend, and it wouldn't be useful for the user as they're most likely only interested in one or two of any given aspect of campus. Pagination needs to be applied to all of our entities.

Request New Garage/Housing/Dining Form

Users would be able to request that a new garage, new housing, or new dining hall location be added to the website. This is beneficial in the case there is a new location for any of these added to UC's campus.

Search & Filtering Functionality

Users should be able to search for specific items to review. This will be important for finding and writing reviews for the entity you're actually interested in.

Takedown Request Form

The takedown request form would allow a user to request an entity be removed from the website, whether that's a review, garage, housing option, or dining hall. This is important when considering the implications of user generated content.

Reply to Reviews

A feature to let reviews be replied to would allow discussions in the review forms, where users could expand and elaborate on their experiences when other users interact with their review.

PWA Functionality

PWA functionality could allow UC Reviews to function smoothly on desktop or mobile devices without the need for a mobile application. It would also enhance performance by using previously loaded offline content.

Report

Users should be able to report other user-generated content that may be misleading or harmful to others.

Post Anonymously

Users might not want to post a review with their name behind it for many reasons, including privacy concerns or freedom to express an opinion that may be unpopular or including sensitive information.

2.10 Problems Encountered and Analysis of Problems Solved

The main problem faced so far has been technical debt. The team has needed to keep up with deadlines for pushing out new features but hasn't been able to allocate time properly to clean up code that may be causing issues now or in the future. Some examples of this that have been faced are not following Angular best practices, not having proper tests, and some problems that have lingered since this project was a proof of concept. Continuing this would result in a buildup of debt that could be difficult to keep track of and cause more issues during development down the road. To work against this problem, the team has taken notes of any occurrences found and created GitHub issues. Doing this allows record-keeping of technical debt and the ability to assign these issues in sprints that aren't as busy as others.

2.11 Recommendations for Improvement

The current state of this project is incomplete, so there are many areas that could be improved upon. The user interface is one of these areas. To improve on this area, user evaluation could be carried out to find specific areas of what could make the user experience better. This could include aesthetics, emotions, user ecology, etc. This would also help to determine whether the application feels fast and if it feels intuitive.

It would be very beneficial to include debt sprints where all members of the team are assigned GitHub issues to cleanup debt. This would allow more debt to be knocked off per sprint, rather than debt issues only being taken care of on the basis of the team's availability

3. CONCLUSION

3.1 Lessons Learned

With multiple people working on the same codebase, the team learned the importance of having set coding standards, naming conventions, code structure, and commenting. This ensures that the code remains readable and maintainable over time. The team improved their skills with git, leveraging branches, cherry picking and resolving merge conflicts which often happened as a result of database migration scripts.

3.2 Abilities and Skills Developed Throughout Project

The team developed a broad range of skills in front-end and back-end development. On the front-end, the team gained experience in Angular and Typescript, building responsive UI, handling user input, and managing client-side state. On the back end, the team gained experience in C# and .NET Core, object-oriented design, and REST API development. The team also strengthened their database skills, implementing a 4th normal form database in MSSQL.

3.3 User Needs Reflection

UC Reviews was developed to meet several user-centered goals identified through an analysis of the target audience: University of Cincinnati students. User needs were analyzed against the backdrop of the five computing processes: selection, creation, integration, evaluation, and administration. For example, in the selection process, we ensured to make it easier for users to find corresponding campus locations or services easily. In creation, we ensured users could provide reviews and feedback in a systematic manner. Integration aimed at consolidating these reviews into a unified dashboard. Assessment prioritized usability and simplicity of features,

whereas administration influenced decisions regarding content moderation and access control.

The primary needs included providing a platform for detailed feedback on campus life, fostering an accessible and user-friendly experience while ensuring privacy and safety. These needs significantly influenced the project's development process.

Throughout the development process, we considered user needs to make key decisions. The interface was intentionally designed to be straightforward, ensuring ease of use for students of varying technical proficiency. Features such as text-based reviews and star rating allowed for nuanced and meaningful feedback, directly addressing the limitations of existing systems like satisfaction kiosks. We also met privacy requirements with features such as anonymous posting and password-free login. These encourage honest and open feedback while reducing the barrier to entry. The design choices supported user goals of being informed, sharing experiences, and encouraging campus change. A password-less login was chosen as it's simpler for users to not have to remember their login.

While we did not formally conduct acceptance testing, the project relied on user personas and scenario-based evaluations to simulate interactions. These methods provided valuable insights into potential usability challenges, guiding the refinement of features and workflows.

3.4 Security Principles and Practices Reflection

The UCReviews team has worked to apply secure coding processes across the application to ensure security and stability as well as the safety of its users. The team expressly adhered to core security principles such as least privilege, fail-safe defaults, defense in depth, and secure-by-design coding. Each of these principles was mapped directly to a practice of the project. Starting at the login page, users are prompted with a password-less login, where all they need to provide is their school email to be sent a temporary password. This login mechanism upholds the principle of fail-safe default by providing temporary access and not reusing passwords. SHA256 with random salt complies with zero-knowledge password storage standards. The temporary passwords use industry standard SHA256 and a random salt to ensure zero-knowledge password storage. The user will be automatically logged out after some time.

The codebase is currently free of any known security vulnerabilities, as the team has worked to ensure there are no outdated or deprecated third-party packages being used. The team also utilizes parameterized queries to prevent SQL Injection to protect their own data from possible outside attackers. There are also no hardcoded credentials, as the team uses environment variables to protect sensitive data. UCReviews features secure coding principles such as failing securely, edge case validation, compartmentalization, and a minimal attack surface. To facilitate defense in depth, environment variables were employed to avoid exposure of sensitive information even if the source code was compromised. Token expiration was also employed to restrict session length and prevent possible abuse. One case of failing securely is a user trying to login with an expired session token, where they will be denied access by the application until they login again. The application follows a microservices architecture, where classes are responsible for specific, actionable items, and do not go out of their own scope. This means that

a failure in one part of the application will not break other parts of the application. Finally, a minimal attack surface is kept. During code reviews, the team makes sure that there are no dependencies or methods in a class that are present but not used.

3.5 Project Conclusion and Future Work

UCReviews enables students to review UC's dorms, dining halls, parking garages, and classes, and successfully provides a platform for sharing valuable insights to the student body. For future work, integrating features such as more advanced searching, a data analytics dashboard for rating trends over time, and personalized recommendations based on user preferences could further enhance user experience. Other students could build on this foundation by expanding the scope to include off-campus resources or leveraging machine learning to analyze review sentiment.

4. REFERENCES AND APPENDIX

4.1 References

SalaryExpert. (2024, November 3). Software Developer Salary in Cincinnati, Ohio.
<https://www.salaryexpert.com/salary/job/software-developer/united-states/ohio/cincinnati>

SalaryExpert. (2024, November 3). Technical Writer Salary in Cincinnati, Ohio.
<https://www.salaryexpert.com/salary/job/technical-writer/united-states/ohio/cincinnati>

University of Cincinnati. (2023, May 9). *HappyOrNot kiosk added on main campus for real-time feedback.* UC News. Retrieved from
<https://www.uc.edu/news/articles/2023/05/happyornot-kiosk-added-on-main-campus-for-real-time-feedback.html>

Laura Kegley. (2024, March 28). Are star ratings enough to gauge customer sentiment? Forbes. Retrieved from
<https://www.forbes.com/councils/forbesbusinessdevelopmentcouncil/2024/03/28/are-star-ratings-enough-to-gauge-customer-sentiment/#:~:text=Without%20accompanying%20context%20or%20detailed%20feedback%2C%20star,on%20the%20star%20ratings%20assigned%20by%20customers.>