E-ISSN: 2707-6644
P-ISSN: 2707-6636
www.computersciencejournals.com/ijcpdm
IJCPDM 2025; 6(1): 37-40
Received: 04-02-2025
Accepted: 06-03-2025

Manan Buddhadev
Rochester Institute of Technology, Rochester, NY, United States

Corresponding Author:
Manan Buddhadev
Rochester Institute of Technology, Rochester, NY, United States

Building trust online: A secure architecture for web portals

Manan Buddhadev

DOI: https://www.doi.org/10.33545/27076636.2025.v6.i1a.112

Abstract
Online transactions have simplified modern life by allowing payments to be made conveniently over the phone or the Internet. People use the Internet for everyday tasks and essential activities, such as sharing personal information when booking a flight ticket or securing an apartment for overseas education. The moment when one has to share anything over the network makes that individual vulnerable to potential information leaks, leading to identity thefts. Most web portals are starting to incorporate secure socket layers into the portals, but many have not, making a user vulnerable to threats. Thus, there is a need for a housing web portal where a user can, without worries, upload confidential data like emails, passport copies, and credit card information. The paper looks into a secure housing website's design considerations, architecture, and implementation. The website contains security features like a virus scan, protection against SQL injections, an encrypted database, data authentication, and logging of the user's keystrokes when logged in.

Keywords: Web security, secure web portal, data privacy, online transactions, sensitive information protection, secure architecture

Introduction

The 21st century has emerged as the age of the Internet, with users relying on it for nearly everything. Many tasks can be accomplished online, from simple activities like looking up synonyms to making mobile phone payments and booking flight tickets. A new thing that has come to light is leasing apartments online, which involves crucial steps like sharing credit card/banking information and your identity proofs, including passports, driver's licenses, or current address proof. The above-mentioned sensitive information is usually of students who don't have anything on their names and have a clean slate identity; if an attacker gets hold of this, they can impersonate that student and create a whole new identity. The application framework focuses on individuals sharing sensitive information online using portals and uploading documents to verify their identity. Therefore, the programmer is responsible for ensuring the website is secure and preventing data leaks to third parties.

The Students Housing application framework provides information for the housing community and the students registered there. It helps maintain the information for housing operations accordingly. Most of the student data is provided by the individual, and a lot of information needs to be secured. The framework consists of the mail, credit card information, work orders, etc., for every student in the housing community.

The fundamental approach to the application's design is to provide secure and least privileged access [12] to all the information by securing the application. The application roles would be admin, receptionist, student, and maintenance staff. Student signs up with an account and can access their mail, credit card information, and personal documents. The receptionist checks for the student's activity and work orders from students. Maintenance staff has access to all the work orders with minimum information about the student. They equip the CIA [10] triads to design secure applications and implement the designed solutions' auditing and access control policies.

The security features [3] explored in this project are Key-loggers, implementation, and prevention, which collect data. At the same time, the user transmits it to the server on the Internet. The application will also incorporate a Secure Socket Layer (SSL) [2] to make the portal more secure. Another security feature planned is data validation, wherein a user cannot enter things like '1111-11-XXXX' for the Social Security Number (SSN), thus ensuring data authenticity.

Furthermore, the application prevents SQL Injection [8] attacks by making the data on the server more secure and not easily penetrable. Moreover, since a user has the right to upload a document, a design consideration is adding a virus scan to the uploaded file so that it does not compromise the central server. Additionally, the application will maintain session [7] information to prevent an attacker from obtaining access to any account a user has logged in to and forgot to log out. Lastly, if there is time at hand after implementing and incorporating the above features, the plan is to encrypt [13] the credit card information and documents received by the user.

Design Considerations

While planning the website layout, considerations regarding the layout, flow, content, and look and feel needed to be made. According to our design, a login page gives access to the profile page containing the user's emails. Users can navigate to the Credit Card Information and Documents uploading section. The website's flow can be inferred from the diagram given below.

Another consideration is the time for session management [7], i.e., if the user is not active for a particular amount of time, the user must be securely logged out from the account. Furthermore, the website currently runs on a *local host*. Hence, it uses the same system resources, which enables us to run scripts that detect key loggers [17]. The website will have a back-end of one relational database with four tables to store the information accordingly. At the same time, the plan is to use encryption [13] on the table to protect the data against vulnerabilities when creating the database table that stores the credit card information. The website allows users to upload files containing personal documents to the server, which creates a potential vulnerability to attacks. To address this issue, a virus scanner will be implemented to reduce the risk of server breaches, provided that users securely log out of their accounts. A block list of file types and known viruses will also be established to prevent unauthorized uploads. If the file uploaded by the user matches any of the values in the deny list, there should be an error, and the file upload should not go through. At the same time, regarding the above design considerations, we need to take a few more steps to make the plan more comprehensive. The steps include incorporating the Secure Socket Layer, which has to be installed on the local host in our case, but if the website is live, the certificates must be purchased, and registrations need to be done. Apart from that, there is a significant focus on issues related to data security by preventing SQL injection, which might give an attacker access to confidential and sensitive data. For this problem, the framework uses parametrized queries as a backup, i.e., if the authentication is breached, the application has encrypted the values in the database.

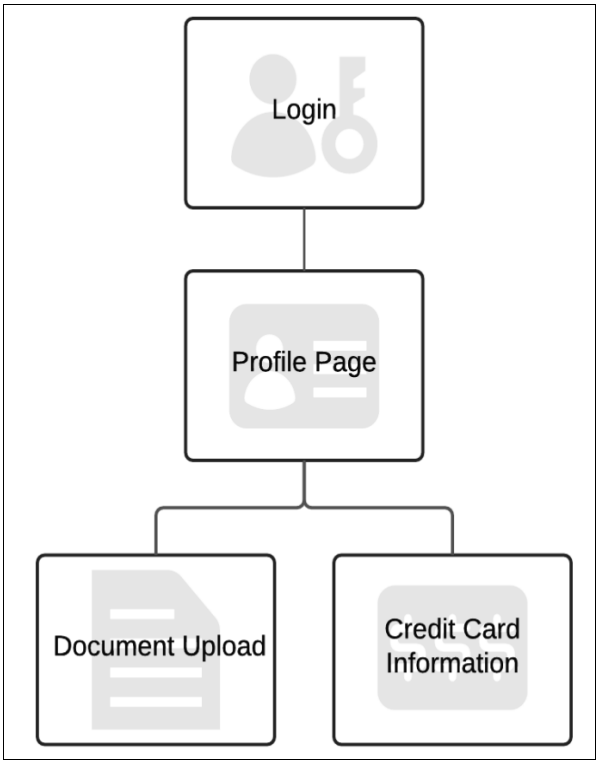

Fig 1: Student housing application flow diagram

Architecture

The project's architecture consists of three pages and one database with four relations. The initial page is the login page, which validates the user's credentials. Depending on whether or not the user was authenticated, the user's access to the profile page will be decided. The user's profile page consists of navigation links to the document upload page and the payment information page, along with a sidebar consisting of emails received by the user.

The database architecture consists of four tables: User, Profile, Credit Card, and Document. The User table contains unique identifiers for each user (UserID) along with their login credentials. The Profile table, linked to the User table through the UserID, includes five attributes: UserID (as a foreign key), Name, Date of Birth, Credit Card Number, and Document ID (as a foreign key).

Credit card information is securely stored in an encrypted format within the Credit Card table, which is linked to the Profile table by the credit card number. This table includes the CVV, expiration date, and credit card number. Additionally, the application stores documents uploaded by users in the Document table, which contains the fields UserID and Document ID.

As an increment of the architecture, the application has used a new database that has the deny list (known viruses and malicious file formats). This database will check if an uploaded file is a potential threat to the website.

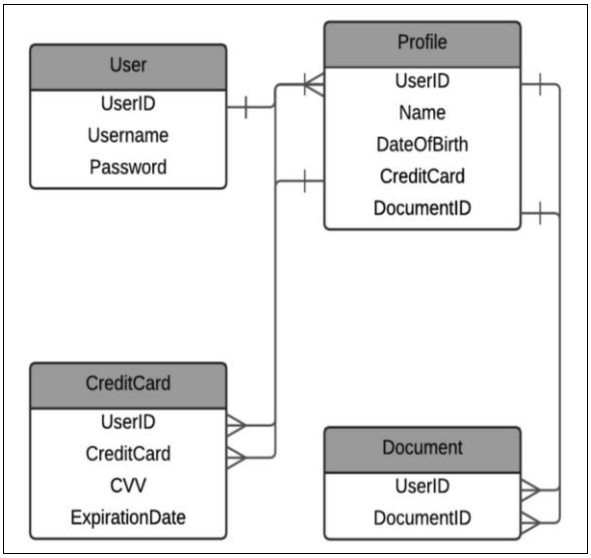

Fig 2: Entity relation diagram of student housing application

Implementation

The application utilizes Meteor [14] for the Student-Housing application framework. The database with the designed tables is stored in Mongo DB [4], which is a feature of Meteor. The database servers were created using Meteor's client-server connection. The database consists of a User Table, Profile Table, Credit Card Table, and Document Table, all implemented based on the Entity-Relationship diagram.

Java implements the application's business logic, while the front end is designed and developed using HTML [6] and JavaScript [5]. The application connects to the database and performs queries through Meteor's client-server protocol. Additionally, the database encryption feature provided by Meteor is utilized to encrypt the Credit Card Table and User Table data. Standard file extensions [1] scan particular files for threats. The flow of the check proceeds as follows: the user uploads a file, the server examines the name and extension of the file, characters after the '.' are split, and a query is executed to verify the extension. If the query returns Null, the file is deemed safe and is added to the User's document database.

Return student File Name. split('.').pop();
Instead of the above code[15], I referred to online forums and came across the code mentioned below [18]

```
If (student File Name. length > 0) {
Var accepted = false;
For (var j = 0; j < all Extensions. length; j++) {
Var current Extension = all Extensions [j];
Var delta Length = student FileName. length – current Extension. Length
If (student File Name. substr (delta Length, current Extension. length). To Lower Case () == current Extension. To Lower Case ()) {
Accepted = true;
Break;
}
}
}
```

The user authentication was initially built using plaintext cipher text encryption, and the team tried SQL injections. Some of the standard SQL injections resulted in successful logins. Therefore, for login details related to user authentication, they decided to use Meteor's built-in package 'accounts-password' [11] for their user table. This package helped encrypt the password, after which none of the SQL injections they attempted were successful. In the current implementation of session management, the username is carried over to the next page, which is tailor-made for that particular username.

Ethical and Legal Issues

The Housing Application caters to students who are gullible and will not think twice before entering the payment details or their identification. The security of such a website, which stores students' personal information and payment details, is critical. If the website is compromised, not only will the person lose a lot of money, but the person will not even be able to stop as attackers have all the details for impersonation. There is a potential threat of identity theft and monetary losses for the user. Since there will be one database administrator(DBA) [16] with the access rights to view the database, with or without encryption, no matter how well the data is encrypted, the users are still vulnerable to attacks. Thus, an ethical issue arises wherein the DBA should not access the client's information until explicitly mentioned to do so by the client who has authenticated themself via the proper protocols. Legal issues arise because the uploaded documents shouldn't be copied to other locations. There can also be client-housing confidentiality wherein none of the information provided by the client can be retrieved through the housing firm. There are various legal issues like data misuse or blackmailing the client by accessing the data the DBA was not supposed to access.

Current Status and Future Work

The website design and development environment is up and running. The databases are created using Mongo DB [4], and all website functionalities such as insert, update, and delete/remove queries are implemented. The team tried different SQL injection attacks [8] on the website's running environment, and since some of them actually worked, they changed the code. During phase 2, they decided to use the 'accounts-password' [11] for the application, which removed all the SQL injection [8] attack vulnerabilities that existed earlier. The application has an up-and-running virus scan that checks for file extensions and rejects suspicious files. The data on the server, along with the login and payment information, is encrypted using Meteor ands, which incorporates data authentication.

The team will try to find a way to validate a file's actual type even if its extension has been changed. Furthermore, they are working on implementing key loggers [9]. To improve the website's security, they will incorporate sessions using packages in Meteor. Additionally, they plan to link emails with the profiles to display them on the profile page. Along with all the previously mentioned work, they plan to run the website on a Secure Socket Layer (SSL) [2] on the local host without buying certificates.

Conclusion

In conclusion, the student housing application framework is a comprehensive solution aimed at securely managing sensitive student data, including identification documents, credit card details, and housing-related communications. By incorporating encryption, session management, SSL, virus

scanning, and protection against SQL injection, the application significantly enhances security and reduces potential risks associated with identity theft and data breaches. It also addresses ethical and legal concerns regarding data privacy and access control, particularly with regard to database administrator privileges. While the application is functional and secure in its current form, future work will focus on strengthening file-type validation, implementing key logger detection, and enhancing session handling. These improvements will further bolster the platform's reliability and safeguard user data in an increasingly digital environment.

References

1. Computer Hope. What are the most common file types and file extensions? [Internet], 2024 Oct 3 [cited 2025 Apr 19]. Available from: https://www.computerhope.com/issues/ch001789.htm
2. Bhiogade MS. Secure socket layer. Available at SSRN 291499, 2002.
3. Chapter OG. Owasp best practices: Use of web application firewalls. [Whitepaper], [Internet], 2008 Jul.
4. Bradshaw S, Brazil E, Chodorow K. MongoDB: The definitive guide: powerful and scalable data storage. O'Reilly Media, Inc., 2019 Dec 9.
5. Flanagan D. JavaScript: the definitive guide. O'Reilly Media, Inc., 2011 Apr 29.
6. Graham IS. The HTML sourcebook. John Wiley & Sons, Inc., 1995 Feb 1.
7. Gutzmann K. Access control and session management in the HTTP environment. IEEE Internet Computing. 2001 Jan;5(1):26-35.
8. Halfond WG, Orso A. AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. InProceedings of the 20th IEEE/ACM international Conference on Automated software engineering, 2005 Nov 7, p. 174-183.
9. Ladakis E, Koromilas L, Vasiliadis G, Polychronakis M, Ioannidis S. You can type, but you can't hide: A stealthy GPU-based keylogger. In Proceedings of the 6th European Workshop on System Security (Euro Sec), 2013 Apr. Citeseer.
10. Merkow MS, Breithaupt J. Information security: Principles and practices. Pearson Education, 2014.
11. Robinson J, Gray A, Titarenco D, Robinson J, Gray A, Titarenco D. Authentication and Deployment. Introducing Meteor. 2015:83-93.
12. Saltzer JH, Schroeder MD. The protection of information in computer systems. Proceedings of the IEEE. 1975 Sep;63(9):1278-308.
13. Silowash G, Lewellen T, Burns J, Costa D. Detecting and preventing data exfiltration through encrypted web sessions via traffic inspection. Techenical Report, CMU/SEI-2012-TN-011, Software Engineering Institute, Carnegie Mellon University, 2013 Mar.
14. Strack I. Getting Started with Meteor. JS JavaScript Framework. Packt Publishing Ltd, 2015 Jun 30.
15. Stack Overflow. How can I get file extensions with JavaScript? [Internet], 2008 Oct 10 [cited 2025 Apr 19]. Available from: https://stackoverflow.com/questions/190852/how-can-i-get-file-extensions-with-javascript
16. Traub JF, Yemini Y, Woźniakowski H. The statistical security of a statistical database. ACM Transactions on Database Systems (TODS). 1984 Dec 5;9(4):672-679.
17. Tuli P, Sahu P. System monitoring and security using keylogger. International Journal of Computer Science and Mobile Computing. 2013 Mar;2(3):106-111.
18. Stack Overflow. Validation of file extension before uploading file [Internet], 2010 Nov 20 [Cited 2025 Apr 19]. Available from: https://stackoverflow.com/questions/4234589/validation-of-file-extension-before-uploading-file