

## ICS491 Assignment 5: SDL: Release

Author: Christopher Nguyen

Date: August 6, 2017

Abstract: Security plays a large role in software development. The Microsoft Security Development Lifecycle (SDL) ensures that software developers understand security basics and the latest developments in security and privacy. Studentals is a database-backed web application that stores sensitive data. To ensure that Studentals complies to the SDL, several requirements and design goals were established for Studentals' implementation. Studentals application was also verified to ensure that it met the set security and privacy tenants before release.

## **1 Introduction.**

Security plays a large role in software development. The Microsoft Security Development Lifecycle (SDL) is a development process that ensures that software developers understand security basics and the latest developments in security and privacy. The SDL requires software developers to receive the proper security training while understanding the proper security and design requirements before implementing a software. Following the SDL will greatly assist organizations decrease the likelihood of exploitable software vulnerabilities and defend against a variety of threats [3]. Failing to follow the SDL may result in the loss of sensitive information from both clients and developers.

The web application that will be developed in compliance with the Microsoft SDL for this and future assignments is Studentals. Studentals (a play on the words “student” and “rentals”) is a Node.js web application to help students and young professionals to find housing or sublets around the nation. Studentals will be a database-backed web application using Javascript, HTML, and CSS. Users can browse available housing or put up their own housing listing on Studentals. This application is essentially a middleman to help students and young professionals ease the grueling housing search process. The sensitive data stored in Studentals are usernames and passwords, user information (such as names and addresses), chat logs (messages between buyers and sellers), and transaction histories. This report will highlight the Requirements, Design, and Implementation phase of the SDL in developing Studentals.

## **2 Requirements.**

The Requirements phase of the SDL is where developers consider what security and privacy concerns and analyze the best ways to create a product that complies with security regulations, costs, and business needs. Security requirements were established, quality bars were

created, and security and privacy risks were assessed in this phase before Studentals entered the Design phase. This section will highlight the Requirements phase of Studentals in compliance with the SDL.

## **2.1 Security Requirements.**

Studentals is a web application that takes in sensitive data in the form of usernames, passwords, names, addresses, and phone numbers from Studentals' users. Alongside this, Studentals also records a history of user chats and user transactions. Defining and integrating security and privacy requirements will allow Studentals to identify key milestones before development. Privacy and security are key factors that can decide whether or not someone uses a certain application.

Privacy is a major concern for all types of users in any web application. In the context of software development and applications, privacy allows users to control, use, and distribute their personal information in the form of data [3]. Software application should inform users of what data is collected, how that data will be used, and give users a choice when their data will be used for outside purposes.

Studentals' users are required to create a profile and input personally identifiable information (PII). PII is any information that can be used to identify an individual. PII includes the names and addresses of an individual—fields enable user to use the Studentals application. To ensure that users have knowledge and control of their data, Studentals will include a Privacy and Agreement page to inform users of what data will be collected. It is also important that Studentals restricts users from viewing other users' PII. With a Privacy and Agreement page in Studentals, users will be informed of how their PII will be used and stored.

Ensuring that private information stays private is also a matter of security. In the context of context of software, writing secure software decreases the likelihood of unauthorized access and theft of sensitive information. Users should not be able to perform operations outside of the context account privileges. This means that users in the client-side should not be able to execute server-side code [3]. By preventing the processing of control instructions or server-side execution, web applications mitigate the risk of losing sensitive data to an attacker.

Any sensitive information from users of any web application should be stored securely. All of the data collected from Studentals, especially PII, will be stored in an encrypted database that no user on the client-side will have access to. By preventing users from accessing the database, attackers disguised as users will not be able to pull the PII of other users [7]. To prevent client-side users from executing server-side code, all forms of data input or text input in Studentals will be sanitized and validated so that no malicious code can attack or steal information from Studentals' databases.

Ensuring that an application is secure in the face of a security flaw is important during the software development lifecycle and SDL. During the course of Studentals' development lifecycle, security flaws will be tracked using Github's Issues feature. Github is a web-based version control system with the ability to manage and track changes in software projects [1]. The ability to create Github Issues during Studentals development will ease the debugging process of any bugs or security concerns that appear during Studentals' development and testing stages. Github Issues also allows users to label the intensity and priority of individual issues by creating tags [3]. With its ability to manage software projects, Github offers a private and secure version control system for Studentals' development lifecycle.

With Studentals collection of PII's and sensitive information, it is essential to inform Studentals' users of their rights to privacy and how their data will be secured and used. Creating a Privacy and Agreement page ensures that Studentals' users understand that their private information will be kept private. Using an encrypted database and preventing malicious server-side code from running on the client-side will prevent unauthorized access of user information. A private Github repository for Studentals will be made to ensure that all of the security issues for Studentals are properly managed and tracked during development.

## **2.2 Bug Bars.**

To ensure quality and security during the software development lifecycle and SDL, it is important to define levels of security and privacy before a software is implemented. These levels and checkpoints are known as "bug bars." Creating bug bars establishes the minimum acceptable levels of security and privacy ensures that the risks associated with security issues are identified so that security bugs can be fixed during development. Establishing bug bars also results in the establishment of security and privacy standards throughout the entire project.

Privacy bug bars are the primary focus on user privacy and administrator privacy concerns. The privacy bug bars for Studentals are divided into end-user scenarios and enterprise administration scenarios. Security bug bars are concerned with scenarios that occur in both the client-side and server-side of Studentals.

### Studentals End-User Privacy Scenarios

Critical	<ul style="list-style-type: none"><li>• PII is collected and stored in a database without an authentication method to prevent unauthorized users from viewing, adding, removing, or editing data.</li></ul>
Important	<ul style="list-style-type: none"><li>• PII is collected and stored in a database without an authentication method to prevent unauthorized users from viewing data.</li></ul>
Moderate	<ul style="list-style-type: none"><li>• PII stored in a database that does not follow a data retention policy.</li></ul>
Low	<ul style="list-style-type: none"><li>• PII is hidden and stored locally as hidden metadata without a way to access, index, or reference said data.</li></ul>

**[Table 1.1 Studentals End-User Privacy Bug Bars]**

### Studentals Enterprise Administrators Privacy Scenarios

Critical	<ul style="list-style-type: none"><li>• Automated data transfer of PII from the user's system without notice in the UI or explicit agreement from the enterprise administrator.</li></ul>
Important	<ul style="list-style-type: none"><li>• Failure to disclose to enterprise administrators the storage or transfer of PII and what database is holding the transferred PII.</li></ul>
Moderate	<ul style="list-style-type: none"><li>• No mechanism provided to help enterprise administrators prevent accidental disclosure of PII.</li></ul>
Low	<ul style="list-style-type: none"><li>• Enterprise administrators are not allowed to create new, nonexistent tables on the database and view hidden metadata.</li></ul>

**[Table 1.2 Studentals Enterprise Administrators Privacy Bug Bars]**

Studentals' commitment to privacy extends to both end-users and enterprise administrators. The bug bars for end-users focuses on ensuring that only authorized Studentals users can see and edit certain types of data in the Studentals' database [Table 1.1]. The bug bars for enterprise administrators focuses on ensuring that administrators are notified of where the data is going and if the data is safely stored [Table 1.2].

### Studentals Server Security Scenarios

Critical	<ul style="list-style-type: none"><li>• The ability to either execute arbitrary code or obtain more privilege than authorized and access unauthorized files.</li></ul>
Important	<ul style="list-style-type: none"><li>• Unauthorized permanent or persistent modification of any user or system data or database configurations.</li></ul>
Moderate	<ul style="list-style-type: none"><li>• Cases where attackers can locate and read information anywhere on the system, especially PII on the database.</li></ul>
Low	<ul style="list-style-type: none"><li>• The console log printing errors (with specific function and REST API references) that are not displayed on the UI and do not interfere with the basic operations of the application.</li></ul>

**[Table 2.2 Studentals Server Security Bug Bars]**

### Studentals Client Security Scenarios

Critical	<ul style="list-style-type: none"><li>• The ability to either execute arbitrary code or obtain more privilege than authorized and access unauthorized files.</li></ul>
Important	<ul style="list-style-type: none"><li>• Cases where an attacker can locate and read information on the system or database that was not intended to be exposed such as system data or database configurations.</li></ul>
Moderate	<ul style="list-style-type: none"><li>• Cases where attackers can find and read program files from the system and their version numbers.</li></ul>
Low	<ul style="list-style-type: none"><li>• Ability for attacker to present a UI that is different but visually identical to Studentals.</li></ul>

**[Table 2.2 Studentals Client Security Bug Bars]**

Studentals' commitment to security deals with issues that can occur in server-side and client-side scenarios. The bug bars for the server-side prioritizes scenarios involving illegal access to Studentals' database and illegal access to any server-side information that would give attackers inside into the digital architecture of Studentals. On the server-side, it is imperative that

all the file permissions and settings on Studentals are hidden from the eyes of attackers. The bug bars for the client-side prioritizes situations where attackers from the client-side can find and view information that is not supposed to be available to them. Having a secure client-side for Studentals can prevent malicious activity running on the server-side and prevent attackers from stealing server-side information.

### **2.3 Security and Privacy Risk Assessments.**

Assessing risks ensures the security and integrity of Studentals as a web application. Studentals' goal is to help students and young professionals find housing and put housing on the market for others to rent or sublet. As a result, the PII that Studentals is required to store on a database include usernames and passwords, user information (such as names and addresses), chat logs (messages between buyers and sellers), and transaction histories. Whenever a user puts a housing available for rent or sublet on Studentals, the data of that housing is also stored on a database as well. A house's data will include the image of the house, a description of the house (along with the rent and sublet length), its address, and its price.

Since Studentals is a Node.js web application, it is imperative that all of the dependencies used to create this web application is updated using the "*npm install*" before the application is run to ensure that the entire application and its dependencies are updated and has the latest security patches. Before a user can browse Studentals, they will see a Privacy and Agreement page that will inform them that their PII and uploaded housing data will be collected and stored in a database.

Organizations that use Studentals can quickly upload their house that is available for rent in the housing listings like a normal Studentals user. Every Studentals user can see which houses are available from other users but they will only be able to edit housing listings which they have



created. Studentals' users will also be able to see their own user data—not the user data of other users. The database that will be used to store data from Studentals will be encrypted and have an authentication mechanism so that attacks will not be able to access private information.

Studentals require that all users read and accept the terms and agreement of Students' data collection policies. Threat modeling is requirement for the parts where a user creates a Studentals account, browses the housing listing, and adds a house to the housing listing for other users to rent or sublet. It is imperative that throughout Studentals' development process, constant security reviews should be done to ensure that the data from Studentals is properly stored in a database and all the dependencies and packages used to develop Studentals is up to date.

### **3 Design.**

The Design phase of the SDL occurs after the requirements are established in the Requirements stage. This phase is critical for establishing best practices around design and functional specifications and performing risk analysis that will reduce security and privacy issue before the Implementation phase of Studentals. This section highlights the design requirement establishment, attack surface analysis and threat modeling that was done for Studentals.

#### **3.1 Design Requirements.**

Studentals is a Node.js web application that uses a combination of the HTML, CSS, and Javascript programming languages. This application also has a secure, encrypted database that will either use a SQL or NoSQL solution to protect user data. Since Studentals collects both PII to generate user profiles, two database tables or collections are required to hold the user profiles and the houses that users put up for rent or sublease. For all aspects of Studentals' development process and development tools, the security and privacy concerns and requirements should be

enforced. During Studentals' development process, all of the code will be held in a private Github repository for issue tracking and project management.

Ensuring that users have control and know where their private information is going is important to any professional web application. Studentals collects PII to generate user profiles and enable users to be able to browse Studentals' housing list and add new houses to it. The houses that users also put up on Studentals' housing list are also saved to Studentals' databases. To ensure that users understand their privacy rights, Studentals will inform its users of Studentals' data collection policies and data retention policy in a Privacy and Agreement page. To prevent issues highlighted in the privacy bug bars in the Requirements phase, the database used to store Studentals' data will be encrypted to prevent outside and unauthenticated access. All of the configuration files and tokens to use the database will also be hidden from unauthorized eyes. As stated in the Requirements phase, private user information will be kept private on Studentals.

Since Studentals exists as a platform to assist in communication and transaction, ensuring that Studentals is secure is a priority. The databases containing the database tables and collections for Studentals must be encrypted and accessible to those with the proper privileges. Normal Studentals users should only be allowed to add, remove, and edit their own user information and their housing in the housing listing—Not anyone else. On the server-side, Studentals database can only be accessed with a correct administrative password and connection Strings. This prevents attackers from editing and changing the keys and configurations of the Studentals database. On the client-side, prepared statements will be used to view, insert, remove, and edit data from the Studentals UI. This is to prevent SQL injections and database tampering from attackers [4]. The client-side should also contain no mechanism for Studentals users to

view the underlying configuration files that power Studentals and its databases. All of the data and data entry points for Studentals must be secure and proper authentication must be done to prevent users from elevating their privileges or to prevent attackers from making unauthorized changes to Studentals' data.

Studentals is required to keep all of user data safe and encrypted. All users of Studentals must be able to understand their privacy rights as users and only be able to access information that pertains to them. Users and attackers should not be able to tamper with the data stored in Studentals' database and view the database configurations from the client-side. In order to ensure security and privacy, Studentals should continually be tested on the client-side to detect any bugs or issues.

### **3.2 Attack Surface Analysis.**

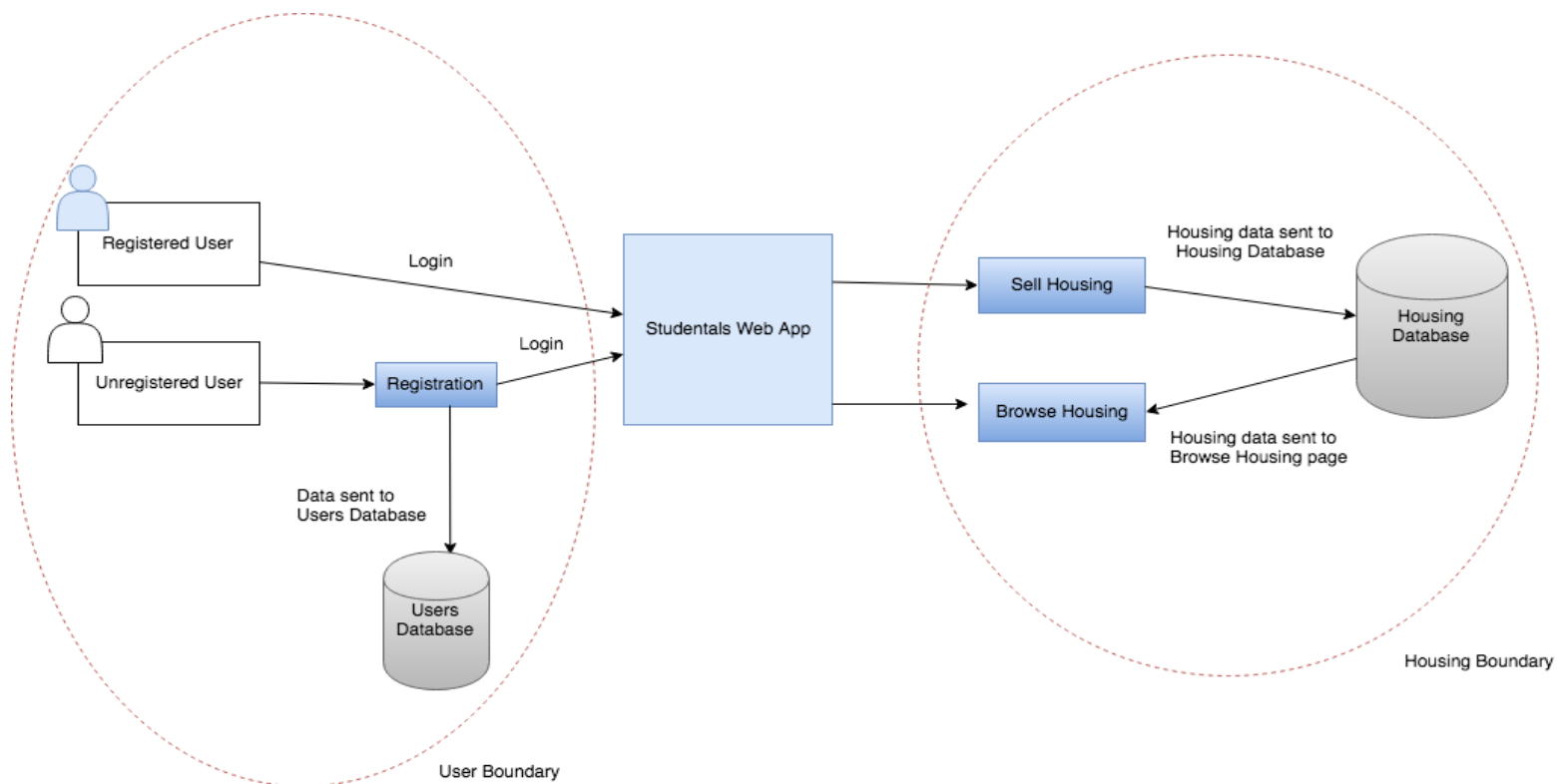
An Attack Surface Analysis is the sum of all possible paths in which an attacker can infiltrate and steal, damage, or alter any available assets. It is used by software developers to identify and manage application security risks as the application is changed in the development process [7]. An application's attack surfaces can be identified by identifying different entry points for attackers from an application's source code.

Because Studentals is a database-backed web application, it has many attack surfaces available. The most foreseeable points of entry for an attack in Studentals is the UI form fields and database. The UI form fields in Studentals will give Studentals' users the ability to input data in the database. By giving users the ability to access the database from a front-end UI, there is a risk that an attacker could steal data from the database or initiate a SQL injection. This is a high-risk area because being able to send data to the database when creating a user account or adding a new house to the housing listing are primary features that Studentals cannot exist without.

In order to prevent attackers from attacking Studentals' databases from the UI form fields, prepared statements must be used to send and store data after the submission of those UI form fields. Prepared SQL statements prevent attackers from executing malicious SQL code that can harm Studentals' databases upon execution. The UI forms should also have regular expressions to validate input values. An attacker having access to Studentals' databases would result in the loss or destruction of both Studentals' user accounts and housing information.

### 3.3 Threat Modeling

Using a structured approach to potential threats in the Design phase assists in identifying security vulnerabilities and risk. Threat Models are used in conjunction with Data Flow Diagrams keep track of the information flow in a program. Threat Models also contain privilege boundaries to establish the privilege levels in a software system.



[Figure 1: Studentals Threat Model]

The Threat Model for Studentals consists of two privilege boundaries—the user boundary and the housing boundary [Figure 1]. Both boundaries are points of access to one of Studentals’ databases. The user boundary consists of user login operations that involve users creating a new account for Studentals and logging in. The database table associated with user boundary is the Users database which contains user account information as well as PII. The housing boundary consist of operations that involve browsing the housing listings and putting a house up for sale. The database table associated with the housing boundary is the housing database which contains housing data uploaded by users. From the Threat Model, it is evidenced that the two boundaries that attackers can strike from are the user boundary and the housing boundary.

#### **4 Implementation.**

The Implementation step of the SDL assists the end user to make informed decisions about secure deployment and the time to establish best practices for detecting and removing security issues from the code. This section will detail the list of secured tools that will be used for Studentals’ development, highlight some depreciated functions in Javascript, and discuss a static analysis tool that will be used during the development of Studentals. Studentals code will be stored on Github and a README.md file will be made to document Studentals’ development history.

##### **4.1 Approved Tools.**

Studentals is a web application that requires a database. The core functionality of Studentals’ code will be written in Javascript with HTML and CSS for Studentals’ user interface. The database code for Studentals will be in SQL—as opposed to NoSQL. Studentals will also use Node.js, a server-side runtime for Javascript which enables the application to be ran using the “node” command. Node.js also includes the Node Package Manager (npm) which assists in

the installation of packages and dependencies. No IDE will be used for Studentals' development as Node.js will allow the application to be tested on a local host through the node command. All the code for Studentals will be written inside a text editor.

The required and approved tools for Studentals' development process is divided into two sections. The first is the standard set of tools required to develop Studentals such as runtime environments and code quality tools [Table 3.1] The second is the node modules that need to be installed with the "npm install" command for Studentals to properly run [Table 3.2].

### **Studentals Development Requirements**

<b>Tool</b>	<b>Minimum Required Version</b>	<b>Comments</b>
JSHint	Version 2.9.0	Javascript code quality and code analysis tool.
MySQL	Version 5.7.19	SQL database that can be connected to Node.js applications using the mysql module.
Node.js	Version 7.3.0	Javascript server-side runtime environment.
Node Package Manager (npm)	Version 5.0.0	Javascript package manager that assists the installation of packages and dependencies.
Sublime Text	Version 3 – Build 3000	Customizable text editor for code, markup and prose.
XAMPP	Version 5.6.0	Apache distribution containing MariaDB, a MySQL database.

**[Table 3.1 Tools Required for the Development of Studentals]**

### Studentals Node Module Requirements

Node Module	Minimum Required Version	Comments
connect-flash	Version 0.1.1	Middleware used in conjunction with the cookie-parser module to display messages on the front-end of login and registration status.
cookie-parser	Version 1.3.5	Module to parse and create cookie information for login authentication.
ejs	Version 2.3.2	Module for templating HTML (used to make a clean front-end with minimal CSS code).
express	Version 4.13.0	Module used to route pages and manage APIs.
mysql	Version 2.7.0	Module to connect MySQL databases to Node.js applications.
passport	Version 0.2.2	Authentication middleware for Node.js applications.
passport-local	Version 1.0.0	Passport strategy to authenticate logins into Node.js applications with a username and password.

**[Table 3.2 Node Modules Required for the Development of Studentals]**

For the standard development tools required for Studentals, it is essential for Node.js and npm to be used in this project because Studentals' architecture requires a server-side runtime. MySQL should be used as the SQL database for Studentals. For this project, XAMPP will be used to access MariaDB, a MySQL database. For the Node.js side of this project, the mysql node module is required to ensure that the application can properly send and display data from the MySQL database connected to Studentals. Passport, a Node.js module, will be used as a middleware to provide the user authentication and user account creation in Studentals. As with most Node.js applications, the express module is required to assist in the routing of webpages and API calls. Although additional node modules may be installed to assist in the formatting of JSON data or

using Regular Expressions, the main two node modules that are crucial for Studentals to run are the express and mysql modules.

#### **4.2 Deprecated Functions.**

Javascript contains a standard library of objects and a core set of language elements such as operators and statements. Deprecated functions are functions that are in the process of being replaced by alternative or newer ones. Deprecated functions should be avoided and their usage sometimes results in errors or flags that suggests alternative functions that achieve the same means. In order for Studentals' code to be safe and in compliance with the latest standards, it is important to note some of the deprecated functions that exist in the Javascript language.

The table of deprecated Javascript functions and statements related to Studentals can be seen in Table 4. A major area for concern are the “for each” loop which is to be replaced with the “for of” loop which achieves the same means of creating a loop that iterates over iterable objects [5]. Date objects should also retrieve years by using the *getFullYear()* function as opposed to *getYear()* [4]. Lastly, when retrieving a value of a param of an Express Request object, the *params(name)* function should be used—not *param(name)* [Table 4].



## Deprecated Javascript Functions and Statements for Studentals Development

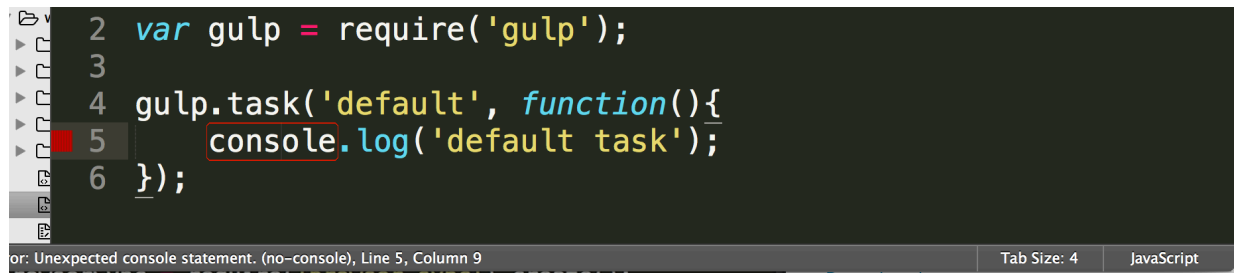
Deprecated Functions and Statements	New/Alternative Function	Return Value
<i>dateObj</i> .getFullYear()	<i>dateObj</i> .getFullYear()	A number corresponding to the year of the given date, according to local time.
for each ( <i>variable</i> in <i>object</i> ) { statement }	for ( <i>variable</i> of <i>iterable</i> ) { statement }	Creates a loop iterating over iterable objects.
<i>req</i> .param(name)	<i>req</i> .params(name) or <i>req</i> .body(name) or <i>req</i> .query(name)	Returns the value of param <i>name</i> when present.

[Table 4 Depreciated Javascript Functions and their alternatives]

### 5 Static Analysis.

A static analysis is an analysis in which source code prior to compilation and run-tests. Static analysis tools provide a scalable method of security code review to ensure that secure coding policies and coding standards are kept during development. During the development of Studentals, the JSHint static analysis tool will be used to lint and detect errors in code [3].

JSHint is an open-source tool that detects errors and potential problems in Javascript code. JSHint can also display the metrics in Javascript code such as the number of functions in a file and the number of statements per function. While code can be copy-pasted and tested on JSHint's online service, JSHint can be installed into a project's repository through the npm or as a plugin for the Sublime Text editor. Since Studentals will be using Sublime Text as the main text editor throughout development, the JSHint plugin will be installed to Sublime Text for Studentals [Figure 2].



```
2 var gulp = require('gulp');
3
4 gulp.task('default', function(){
5   console.log('default task');
6 });
```

or: Unexpected console statement. (no-console), Line 5, Column 9

Tab Size: 4 JavaScript

[Figure 2: JSHint on Sublime Text]

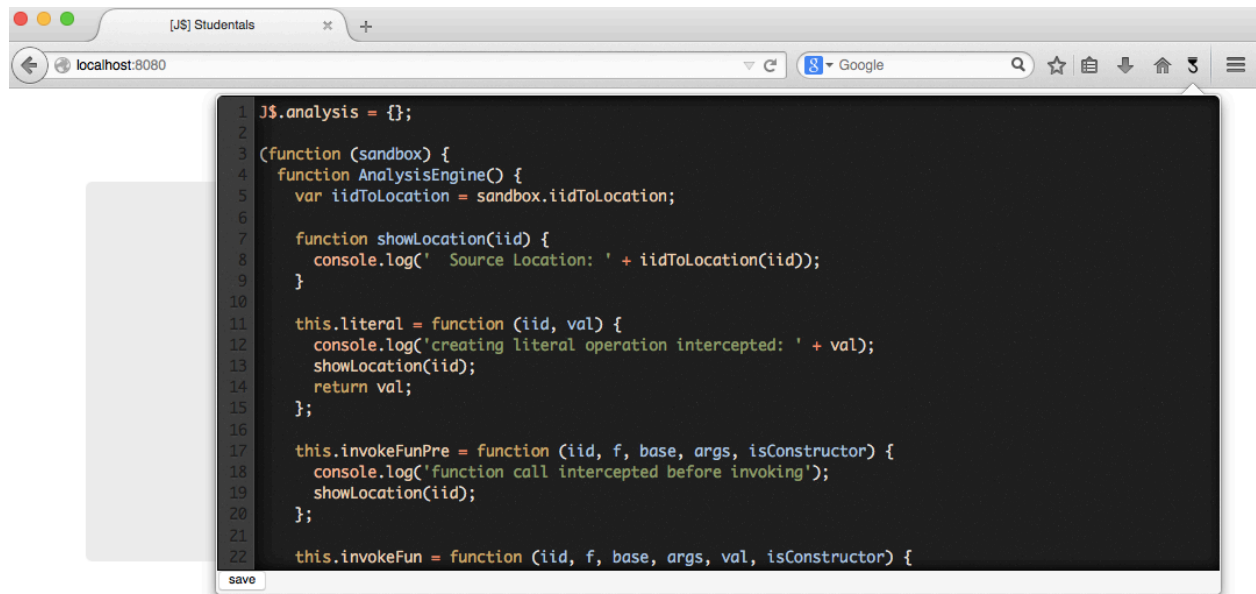
JSHint is a reliable code quality tool that can perform static analysis of Javascript code. Because JSHint can be installed into a variety of text-editors and project repositories, it is an industry-approved static analysis tool [3].

## 6 Verification.

The Verification phase in the SDL ensures that the security and privacy tenets in the Requirements phase and the Design phase are met. This phase involves performing dynamic analysis of code, performing fuzzy testing, and an attack surface review. The application of the Verification phase to Studentals ensured that Studentals is compliant with the latest security updates and the standards set in the previous phases of the SDL.

### 6.1 Dynamic Analysis

Dynamic Analysis of code involves performing runtime verification of software checks using tools that monitor an application's behavior for memory corruption, user privilege issues, and other potential security problems. Since Studentals is a Node.js application programmed in Javascript, the dynamic analysis of Studentals' code was done using Jalangi, an open-source dynamic analysis framework for Javascript that can be ran as a Firefox extension [2]. Jalangi was developed at the University of California, Berkeley, as a framework for both front-end and back-end Javascript. Any browser tab with Jalangi enabled is put into a sandbox environment where the Javascript code of that tabs contents can be analyzed.



**[Figure 3: Jalangi used on Studentals]**

Although Jalangi provides users with the ability to write their own custom tests, the default tests and code provided with Jalangi’s documentation website [3] were used to analyze Studentals. By default, Jalangi intercepts and transforms every line of Javascript code loaded by the browser to expose hooks in the program. When used with Studentals locally on the Firefox browser, Jalangi was set to intercept function calls before and after invocation, intercept get field operations, intercept variable read and write operations, and intercept binary operations.

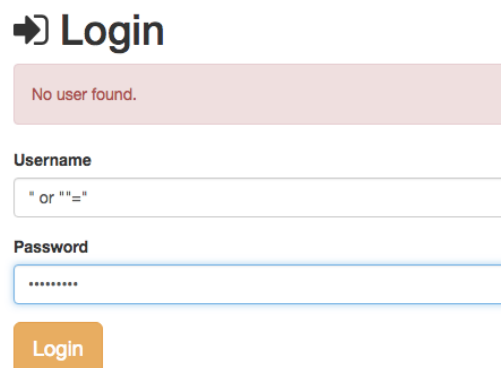
After using Jalangi several times on Studentals during development, it was found that upon logging into Studentals from Studentals’ login page, the Javascript alert that welcomed Studentals’ users to Studentals’ main page could be intercepted. As a result, that “alert(‘Welcome!’)” code was deleted from the Studentals’ main page as it deemed insecure by Jalangi. Since Node.js web applications can be run on the browser, the combination of Firefox and Jalangi were fairly easy to use due to Jalangi’s built-in interception code. Upon using both Google Chrome’s and Firefox’s code inspection tools, it was shown that the server-side Javascript code on Studentals was hidden from the client-side—preventing attackers from seeing

any sensitive user information or information regarding how Studentals' front-end linked with the back-end Studentals database.

## 6.2 Fuzz Testing.

Fuzz Testing is the act of purposely inducing program failure by inserting malformed or random data into an application to reveal potential security flaws prior to release. Fuzz Testing can be done on Studentals' various data-entry points such as the login page, signup page, and sell page. Studentals is a database-backed application that uses a MySQL database, as a result, various SQL injection attempts were performed on Studentals [10].

The login page of Studentals enables Studentals users to login to Studentals if they have a Studentals account. The Studentals user account information is saved into the Studentals database in the Users table. A common SQL injection attack involves entering " or ""=" into both the username and password input fields of an application to allow a hacker to get access to the usernames and passwords in a database [10].



The screenshot shows a web form titled "Login" with a right-pointing arrow icon. Below the title is a red error message box that says "No user found." The form has two input fields: "Username" and "Password". The "Username" field contains the text '" or ""=' and the "Password" field contains several asterisks. Below the input fields is an orange "Login" button.

**[Figure 4: A Failed SQL Injection attempt on the Studentals Login page]**

This SQL injection attempt failed due to the Passport node module. The Passport node module was used as the authentication middleware in Studentals. In passport.js, the Javascript file containing the authentication middleware for Studentals, the SELECT statement for the SQL query has a "?" character in it to escape any user provided data during the user authentication

process. For the MySQL node module, the “?” character in a prepared statement serves a placeholder for values to be escaped. Escaping user input before executing the query is a common method of prevent SQL injections from occurring and it is a simple operation with the MySQL node module [10]. If this attack were to succeed, the usernames and passwords for Studentals’ User data table would be revealed and an attacker would be able to access all of Studentals’ user information.

The Sell page of Studentals allows Studentals users to add a housing onto the housing database for other Studentals users to rent or sublet. This is done when a user completes the form in the housing page and submits it. Upon form submission, an INSERT query is done to insert data to the housing table in the Studentals database. Another common SQL injection attack is to drop and destroy a database table using “DROP TABLE” along with the table name—as a result, “; DROP TABLE housing” was entered and submitted from the Sell page.

```
app.post('/sell', isLoggedIn, function(req, res) {  
  var connection = mysql.createConnection({  
    host: dbconfig.connection.host,  
    user: dbconfig.connection.user,  
    password: dbconfig.connection.password,  
    database: dbconfig.database  
  });  
  var data = req.body;  
  connection.query('INSERT INTO housing SET ?', data, function(err, res) {  
    if (err) throw err;  
    connection.end();  
  });  
  res.redirect('/browse');  
});
```

**[Figure 5: The Insert Statement for Studentals’ Sell Page Submission]**

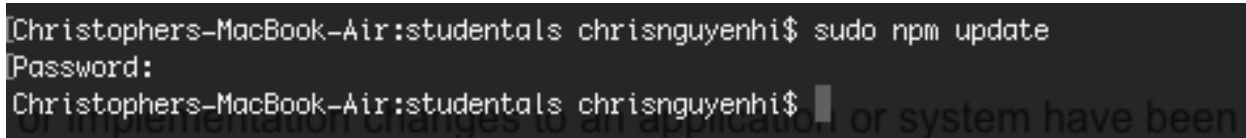
This SQL injection attempt to drop a table did not work due to the “?” character acting as an escape character in the prepared statement [Figure 5]. Several variations of this Drop Table attack were also tested on the Login page and failed to drop the Users and Housing database

tables. If this attack were to succeed, an attacker would be able to remove all of Studentals' database tables and destroy the application.

A similar SQL injection attack was done to drop all of the tables in Studentals with the SQL command “; DROP ALL TABLES;” entered to all of the user input fields and text areas in Studentals. As with the previous attempts the “?” character acting as an escape character prevented this SQL injection attack from succeeding. If this SQL injection attack were to succeed, all of the tables in Studentals would have been dropped and the entire database would have been empty.

### **6.3 Attack Surface Review.**

During the Verification phase of the SDL, another Attack Surface Review is done to ensure that any design or implementation changes to the application have been taken to account and that any new attack vectors created as a result of the changes have been reviewed and mitigated. For Studentals, the time between the Design phase and the Verification phase was not long and as a result, no large version changes to the application's dependencies and tools have been made.



```
Christophers-MacBook-Air:studentals chrisnguyenhi$ sudo npm update
Password:
Christophers-MacBook-Air:studentals chrisnguyenhi$
```

**[Figure 6: Using “npm update” command on Studentals during Verification phase]**

This was confirmed using the “npm update” command on the Studentals application folder [Figure 6]. Since there was no output from the “npm update” command, all of the dependencies in Studentals are up-to-date.

## **7 Release.**

The Release phase focuses on readying a project for public release. To ensure that a project is ready for public release, the development team must plan ways to effectively perform post-release servicing tasks and be ready to address potential security and privacy vulnerabilities. For this phase, an incident response plan was made for Studentals and a final security review was conducted before Studentals' certified release and archive was done.

### **7.1 Incident Response Plan.**

An incident response plan is a plan of action that is executed when a future threat occurs. Studentals' incident response plan involves a privacy escalation team, a group email, and a set of procedures that the Studentals team will follow in the event of a security breach or a privacy breach. Studentals' incident response plan follows privacy escalation response framework (PERF), a systematic process that is used to resolve privacy escalations efficiently.

The Studentals privacy escalation team includes one developer that plays the role of an escalation manager, a legal representative, a public relations representative, and a security engineer. Studentals' escalation manager represents Studentals and Studentals' privacy and business experts. The Studentals escalation manager handles the privacy escalation response process by evaluating the contents of a privacy escalation to determine whether or not more information is required and takes the appropriate actions to proceed working with the reporting party and other contacts. The Studentals legal representative handles any legal-related issues involving the affected parties. The legal representative of Studentals primarily focuses on handling any lawsuits that target Studentals or issuing lawsuits against any party that may have harmed Studentals and Studentals' users. Studentals' public relations representative handles any public relations concerns with any media that requests more information regarding the affiliated

incident. In an incident, it is up to the Studentals public relations representative to relay what caused the incident to the public and ensure that the public understands what steps they can take to mitigate or prevent the incident. Lastly, the Studentals security engineer ensures that the application is secure and that any backdoors or privacy problems in the Studentals application is resolved quickly before attackers can exploit them. In the case of a privacy incident, Studentals' security engineer must pinpoint the cause of the incident and push a new security update to address those causes. In the case of an incident, the lead developer for the Studentals project can be contacted at [cnguyen7@hawaii.edu](mailto:cnguyen7@hawaii.edu).

When Studentals is contacted with an email containing the privacy issue, it is up to the escalation manager to decide whether or not more information is needed. The escalation manager evaluates the content of the escalation to determine the source of the escalation, the impact and scale of the escalation, the validity of the incident, known facts of the incident, timeline expectations to resolve and address the incident while determining the employees that know about the situation or service. After disseminating this information, the escalation manager should contact the appropriate parties to seek a resolution.

During an incident, the appropriate resolutions should be determined with the Students reporting party including the internal incident management, communications and training, human resources actions, as well as external communications. External communications involve online articles, public relations outreach, breach notifications, documentation updates, and service changes—all of which can be handled by the Studentals' public relations representative. After all the appropriate resolutions are in place for any incident regarding Studentals, the Studentals privacy escalation team will evaluate the effectiveness of their privacy escalation response to ensure an effective remediation.



## **7.2 Final Security Review**

Before release, Studentals conducted its final security review (FSR) based on the previously established threat model and code analysis tools as well as the quality gates discussed in the Design and Requirements phase. After running the Studentals application as a client-side user and going through the requirements highlighted in the Design and Requirements phase, it was found that Studentals was free from SQL injections and memory leaks. The latest version of Studentals also passed the JSHint checks and Jalangi tests. There were also no Javascript console logs in the front-end browser that showed any extremely sensitive data for attackers to use for malicious purpose. As a result, Studentals passed the FSR.

## **7.3 Certified Release & Archive**

Studentals currently runs locally through a localhost. The Studentals code is currently stored on Github at <https://github.com/chrisnguyenhi/studentals>. A Github Wiki was also made for Studentals before it was released on: <https://github.com/chrisnguyenhi/studentals/wiki>

## **8 Conclusion.**

The Microsoft SDL ensures that software developers understand security basics and the latest developments in security and privacy. Studentals is a database-backed web application that stores sensitive data as well as PII. To ensure that Studentals holds a fine standard of privacy and security, several requirements and design goals were established. The development of Studentals uses industry-standard tools and code analysis tools to ensure that Studentals is in compliance with the latest security trends and security updates. Studentals databases must be secure to the point where attackers cannot run malicious code that can damage or steal data from Studentals. During the Verification phase of Studentals, all of the Studentals input fields were fuzzy tested with SQL injections and an attack vector review was done to ensure that Studentals is security

compliant and safe for commercial and private use. Studentals was successfully released and is backed up with an incident response plan if the application were to face any security or privacy issues.

## References

1. Github. 2014 April 7. *Mastering Issues*. [Online]. Available:  
<https://guides.github.com/features/issues/>
2. Jalangi. n.d. *Jalangi Online Demo*. [Online]. Available:  
[http://people.eecs.berkeley.edu/~gongliang13/jalangi\\_ff/](http://people.eecs.berkeley.edu/~gongliang13/jalangi_ff/)
3. JSHint. n.d. *JSHint*. [Online]. Available: <http://jshint.com/install>
4. Microsoft. n.d. *SDL Process: Design*. [Online]. Available:  
<https://www.microsoft.com/en-us/SDL/process/design.aspx>
5. Microsoft. n.d. *SDL Process: Requirements*. [Online]. Available:  
<https://www.microsoft.com/en-us/SDL/process/requirements.aspx>
6. MongoDB. 2017. *What is MongoDB?* [Online]. Available:  
<https://www.mongodb.com/what-is-mongodb>
7. Mozilla Developer Network. 2017 June 23. *Date.prototype.getyear()*. [Online]:  
Available:  
[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Date/getYear](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/getYear)
8. Mozilla Developer Network. 2017 March 6. *For Each... In*. [Online]. Available:  
[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for\\_each...in](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for_each...in)
9. OWASP. 2015 July 18. *Attack Surface Analysis Cheat Sheet*. [Online]. Available:  
[https://www.owasp.org/index.php/Attack\\_Surface\\_Analysis\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet)
10. OWASP. n.d. *Testing for SQL Injection (OTG-INPVAL-005)*. [Online]. Available:  
[https://www.owasp.org/index.php/Testing\\_for\\_SQL\\_Injection\\_\(OTG-INPVAL-005\)](https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005))