**Dental Appointment System Client-Server**

1. **Description of secure software techniques incorporated**

|  |  |
| --- | --- |
| **Secure software practice incorporated** | **Incorporated at which part of the application** |
| **Input Validation**  Input validations had been done whenever the browser requires the user to key in any inputs. | Handler functions in the server package like *MakeAppt(), List(), EditAppt(), Browse(), SearchAppt()* where the code will take in date and time input, which need to be an integer value.  The standard *strconv.Atoi()* is used to check that the user input is in the correct format. If the input is not in the required format, an error or panic will occur. |
| **Sanitization of Input**  Sanitization is the process of removing or replacing submitted data from the user. This is done to escape special characters such as “<”, “>” etc so that the html tags are removed to prevent from switching to execution context. | Go templates (html/template) are used in the application to change the display of content according to the context. In this case, it will be escaping content that are HTML so that the tags will be removed before sending to the backend for some execution. This can also help to defend against cross-site scripting attacks as it will remove the script tag prevent any malicious script from sending in.  Go templates are used to display content in all the handler functions in package server like *MakeAppt(), List(), EditAppt(), Browse(), SearchAppt(), Login(),Logout(), Index(), Signup()* as html. |
| **Proper Error Handling and Logging**  Error handling involves capturing and handling of errors in application logic.  Logging allows identification of all operations that occurred. This can help to the owner to keep track of what activities occur behind the scene and can notify them of any suspicious activities. | All the function methods of data structure in the *datatype* package will return error message if there is any error. A ‘*defer’* anonymous function for panic recovery is also declared at beginning of the functions in *userpackage* package as well as server package where the core part and server of the application are. Error handlings are also used in the server side where invalid input are keyed in and error message will be shown to the user on the browser on the error and how they should key in the input for the respective variable (int or string and range if there is any).  Loggings are mostly done in the server package to keep track of all the activities in all the handle functions like *MakeAppt(), List(), EditAppt(), Browse(), SearchAppt(), Login(),Logout(), Index(), Signup()* in the *server* package.  3 custom loggers (in the *ultis* package) had been declared:   1. *traceLog* – trace all activities such as which user log in/log out as well as making or editing any appointment and write to *“trace.txt”* file 2. *warnLog* – keep track of activities that has error or panic and write to *“warn.txt”* file 3. *errLog* – keep track of activities that will critically affect the application and write to *“err.txt”* file |
| **Cryptography Hashing & Encryption**  Hashing turns the original source data into a string or number generated by a hash function that is not reversible.  Encryption, on the other is reversible and can be decrypted using the correct decryption function. This is done by using asymmetric cryptography which make use of a pair of keys (private and public) | HTTPS is used to establish the connection between client and server so that the data sent will be encrypted at the sender side while data need to be decrypted at the receiver side as well using a public and private key respectively to prevent any MITM attacks.  The password for logging in to the account will be hashed (by using an external package called *bcrypt*) and stored in the database so each subsequent log in, the hashed password will be compared to authenticate the user. |
| **Session Management**  A new session (cookie unique value) should always be generated upon new log-in. The cookie parameter Expire or MaxAge should be set to enforce periodic termination of session to prevent session hijacking.  Session identifiers or sensitive information like username and password should only be passed as POST method so the information will not be shown in the URL.  To prevent the session from Man-in-the middle (MITM) attacks, the best practice is to use HTTPS in all requests. | For every new log-in or sign up by the users, a new session (cookie unique identifier) will be set up. The cookie *MaxAge* parameter is set to *1800* which means the session will only be valid for 30minutes to enforce periodic termination of session to prevent session hijacking.  In addition, HTTPS is used to establish the connection between client and server so that the data sent will be encrypted at the sender side while data need to be decrypted at the receiver side as well using a public and private key respectively to prevent any MITM attacks. |

1. **Description of Idiomatic Go Techniques**

|  |  |
| --- | --- |
| **Idiomatic GO Techniques** | **Incorporated at which part of the application** |
| **Commentary**  GO provides block comments (/\* \*/) and line comments (//). Block comments are norm and are used throughout the code and package.  Every package should have a package comment before the package clause to introduce the package and provide information relevant to the package. | In every package, there will be a comment before the package clause to introduce on the package and provided information relevant to the package.  Line comments are used throughout the code as it the norm for commenting in GO. |
| **Naming Convention**  The visibility of a name (variable/function) outside a package is determined by whether its first character is an upper case. If the first character is an upper case, it means it can be exported and use in other package while lower case only can be used within own package.  By convention, all the package name should be lower case and no underscore or mixed caps to be used. Hence, all the package name are in lowercase and only the functions or variables that are needed by other packages have their first letter in upper case.  Naming of variables should be as intuitive as possble | For every package used in the application, the exported functions and variables have their first letter in uppercase to show that they can be used in other packages.  For the local variables to be used within the package, camel-case convention is used for 2 or more words.  All the variables or function declared in the packages are intuitive. |
| **Function & Blank Identifier**  GO’s functions and methods can return multiple values such as the intended results as well as an error if there is any.  In GO, programmers are encouraged to check for any error and handle it before writing more codes. Blank identifier can also be assigned to any return value from a function if the value is not needed for executing the subsequent code, this can help to save memory as a dummy variable will not be needed just to contain the result. | All the function methods of data structure in the *datatype* package as well as functions in the *userpackage* package will return its intended results and error message if there is any error.  Blank identifiers are also used when the return values is no needed for executing the subsequent code for eg in *SearchAppointment()* in admin.go, *printAvailDentist()* in patient.go etc. |
| **Defer statement**  GO’s defer statement will defer any function to the end of the code before executing it. This is an effective way to help the programmers release resources when they are not need for eg close a file after opening a file, close a connection after query data from the database etc, which we can write right after we open a file to prevent us from forgetting to release them afterwards and this might cause problem to the application if it is not done. | *Defer* statements are used for panic recovery as well as closing of files.  A ‘*defer’* anonymous function for panic recovery is also declared at beginning of the functions in *userpackage* package as well as *server* package where the core part and server of the application are.  A *defer* statement for closing of file is used in the *utils* package for *TraceLogging(), WarnLogging and ErrorLogging().* |
| **Concurrency**  In GO, while dealing with concurrency, it is encouraged to use channel for communication between different goroutines. The communication in channel is bidirectional by default which means you can send and receive values from the same channel. Moreover, channels send and receive until the other side is ready which allows goroutines to synchronize without explicit locks | Goroutines are used when patient had keyed in the required details for making or editing appointments. 3 goroutines are executed concurrently to update the time slot as well as adding the appointment to the appointment list and dr appointment list. Channels are used to send and receive data. Buffered channel is used for to synchronize between the goroutines to avoid racing conditions.  A buffered channel of 1 Is used for the gouroutines for *AddtoApptList()* and *UpdateAppt()* in *patient.go* under the *userpackage* package. |

1. **GO Documentation**

The GO documentation for the application can be found in the GO DOC folder where the PDF can be found for all the package used in the application.