**Coding Guidelines & Best Practices**

***Coding Guidelines:***

Following are some of the coding Conventions used:

1. We have created application in JSP & servlets. We have used Model-View-Controller architecture in which the model manages data of the application, view is responsible for providing the outside world with the means to interact with the program's client data. The controller receives user input and makes calls to model objects and the view to perform appropriate actions.
2. Java files have following format:

-Beginning Comments

-Package & import statements.

-Class or interface declarations.

1. Naming Conventions:
2. All package names must be in lowercase letters. For e.g. mypackage, user etc.
3. Class Names must start with a capital Letter & should be meaningful for e.g. Student, Login etc.
4. Variable names must be in mixed case starting with a lower case. E.g. timeslot, appointment etc.
5. Constant Names must be in uppercase letters & the words are separated by underscore. E.g. MAX\_VALUE, MIN\_VALUE.
6. Method names must be verbs & written in mixed case starting with lowercase. E:g getTimeSlots(), getStaffName() etc.
7. Getters & setters are used to access the data. Data items are made private members to ensure data hiding.
8. File Extensions
9. Java source files must have ‘.java’ extensions for e.g. EmailNotifier.java.
10. JSP files must have extension ‘.jsp’ for e.g. Admin.jsp, AddUser.jsp.
11. Class names must be declared in separate files with the same name as the file name.
12. The imported classes must be explicitly listed for Example

**import java.util.ArrayList**; //Not import java.util.\*;

1. Variables must always be initialized when declared.
2. Code must be properly commented. Use ‘//’ for all non-javaDoc comments.
3. JavaDoc Comments must have following format:

***/\*\****

***\* Returns array of TimeSlots of a staff member.***

***\****

***\* @param String Email-id of the staff***

***\* @return Array of TimeSlots***

***\* \*/***

***public ArrayList<String> getTimeSlots(String sEmail)***

***{***

***...***

***}***

1. **Design Pattern:** *Factory Design Pattern* in many places so that it is easy to extend the classes in future by just adding a class to make it work with no modification.
2. Error handling is done explicitly wherever it is needed. We have used MySQL as database & have used try & catch the SQL exception wherever application interacts with the database.
3. We used GIT for code version control & repository.

***JSP Best Practices:***

Following are some of the best practices followed during development of web application in JSP,

* Separate HTML from Java
* Place business logic in java class
* Use an appropriate inclusion mechanism
* Use stylesheets
* Use the MVC pattern
* Use available custom tag libraries
* Determine the appropriate level of XML compliance
* Use JSP comments in most cases
* Follow HTML best practices