	Weight (%)	N (0-4.9)	P (5 - 5.9)	C (6 - 6.9)	D (7 - 7.9)	HD (8 - 10)
		Unacceptable Fails to identify what is required	**Basic** Reflects the beginnings of understanding what is required	**Expected** Basic understanding and delivery of what is required.	**Good** Reflects a mastery of what is required	**Excellent** Reflects the highest level of performance, beyond what is required
SOFTWARE DESIGN (DETAILED CLASS DIAGRAM)						
A. Detailed Class Diagram	10%	-Failed to attempt detailed class diagram Or - Class names and stereotypes have a lot of mistakes or are missing - Majority of boundary class and control class are identified wrongly - Majority of attributes of class has mistakes - Majority of the data types and visibility modifiers of the attributes do not make sense - Majority of methods and their visibility modifiers of the classes have mistakes - Major mistakes (or missing) parameters and return types of the methods - Majority of relationships (and their navigation) and mulpicity between classes are identified wrongly - The detailed class diagram has not consistently followed Java or Python coding conventions	- Class names and stereotypes have most significant mistakes or are missing - Most of boundary class and control class are identified wrongly - Most of attributes of class has mistakes - Most of the data types and visibility modifiers of the attributes do not make sense - Most of methods and their visibility modifiers of class have mistakes - Most mistakes (or missing) parameters and return types of the methods - Most of relationships (and their navigation) and mulpicity between classes are identified wrongly - The detailed class diagram has not consistently followed Java or Python coding conventions	**- Class names and stereotypes have some mistakes or are missing** **- Some of boundary class and control class are identified wrongly** **- Some of attributes of class has mistakes** **- Some of the data types and visibility modifiers of the attributes do not make sense** **- Some of methods and their visibility modifiers of class have mistakes** **- Some mistakes (or missing) parameters and return types of the methods** **- Some of relationships (and their navigation) and mulpicity between classes are identified wrongly** **- The detailed class diagram has not consistently followed Java or Python coding conventions**	- Class names and stereotypes have a few mistakes or are missing - A few of boundary class and control class are identified wrongly - A few of attributes of class has mistakes - A few of the data types and visibility modifiers of the attributes do not make sense - A few of methods and their visibility modifiers of class have mistakes - A few mistakes (or missing) parameters and return types of the methods - A few of relationships (and their navigation) and mulpicity between classes are identified wrongly - The detailed class diagram has consistently followed Java or Python coding conventions	- Class names and stereotypes have no mistakes and all the required classes are identified - All the boundary class and control class are identified correctly - All the attributes of class are correct - The data types and visibility modifiers of the attributes make complete sense - Methods and their visibility modifiers of class are completely correct - Parameters and return types of the methods are correctly identified - Relationships (and their navigation) and mulpicitiicies between classes are identified correctly - The detailed class diagram has consistently followed Java or Python coding conventions
IMPLEMENTATION						
B.Implementation **Following criteria will be considered** **1.1 Executing the product** **1.2 Level of completeness** **1.3 Consistency across all the featues** **1.4 Informative Feedback** **1.5 Prevent errors** **1.6 User has control of the application** **1.7 Code documentation** **1.8 Code quality**	75%	1.1 Product failed to launch Or 1.1 It is not runable on mentor's computer 1.2 Most of the features are incomplete or missing major functionalities 1.3 Inconsistent design of the menus and representation of the data 1.4 User is provided with informative feedback only a very few times 1.5.1 No appropriate error messages are displayed for any exception and user introduced errors 1.5.2 Functionalities have numerous logical errors 1.5.3 Most of the functionalities have high chance of crashing 1.6.1 When mentors check your product, he/ she do not know where to start 1.6.2 User can not go back to the main menu whenever desired 1.7 Source code is not documented or documented poorly with auto generated comments from IDE 1.8.1 Source code is not at all optmised, has redudant code and/or hard coded data values 1.8.2 High dependency between the code	1.1 Product launches but with some troubleshooting 1.2 Most of the features are somewhat complete or missing some major functionalities 1.3 Some inconsistent design of the menu and representation of the data 1.4 User is provided with informative feedback a few times 1.5.1 A few appropriate error messages are displayed for any exception and user introduced errors 1.5.2 Functionalities have quite a number of logical errors 1.5.3 Most of the functionalities have medium chance of crashing 1.6.1 When mentors check your product, he/she feels some control of the product 1.6.2 A few times, user can go back to the main menu whenever desired 1.7 Source code is somewhat documented or somewhere documented poorly with auto generated comments from IDE 1.8.1 Source code is not very optmised, has redudant code and/or hard coded data values 1.8.2 Dependency is quite high in the code	**1.1 Program launches with no issues** **1.2 Some features are incomplete or missing minor to some major functionalities** **1.3 Some consistent design of the menus and representation of the data** **1.4 User is provided with informative feedback at some times** **1.5.1 Some appropriate error messages are displayed for any exception and user introduced errors** **1.5.2 Functionalities have some logical errors** **1.5.3 Most of the functionalities have low chance of crashing** **1.6.1 When mentors check your product, he/ she has fair control of the product.** **1.6.2 Sometimes, user can go back to the main menu whenever desired and with some impact to the data** **1.7 Source code is fairly documented and with minimal to no auto generated comments from IDE** **1.8.1 Source code is fairly optmised, has some redudant code and/or hard coded data values** **1.8.2 Dependency of the code is visible and have not used OOP to its benefit**	1.1 Product launches with no issues 1.2 Most of the features are complete 1.3 Majority of the design of the menu and representation of the data are consistent 1.4 User is provided with informative feedback at most of the times 1.5.1 Majority of the error messages are displayed for any exception and user introduced errors are appropriate 1.5.2 Functionalities have minimal to no logical errors 1.5.3 Functionalities have minimal to no chance of crashing 1.6.1 When mentors check your product, he/she has good control of the product 1.6.2 Majority of the time user can go back to the main menu whenever desired and with no impact to the data 1.7 Source code is documented for all the files with inline comments included 1.8.1 Source code is optmised, has minor redudant code and no hard coded data values 1.8.2. Moderate dependency between the code and can be some benefits of OOP implemented, however, can be improved	1.1 Program launches with no issues 1.2 All of the features are complete 1.3 Consistent design of the menus and representation of data (and features) 1.4 User is provided with informative feedback at all times 1.5.1 Appropriate error messages are displayed for any exception and user introduced errors 1.5.2 Functionalities have no logical errors 1.5.3 Program does not crash 1.6.1 When mentors check your product, he/ she has an excellent control of the product 1.6.2 User can easily go back to the main menu whenever desired and with no impact to the data 1.7 Source code is properly documented with inline comments included where appropriate. 1.8.1 Source code is optmised, has no redudant code and no hard coded data values 1.8.2 Code dependency is arguable but can be accepted to some degree. Use of OOP is evident.
C. Mapping Detailed Class Diagram to product's actual architecture	5%	- Detailed class diagram does not map to the system at all	- Very little part of detailed class diagram maps the system	**- Detailed Class Diagram mostly maps to the system**	- Detailed Class Diagram mostly maps the system	- Detailed Class Diagram completely maps the system
D. User Manual	10%	No user manual Or - User manual is not formatted properly - It takes a lot of time to understand the user manual - It does not have any screeshots to guide through the manual - By following user manual, the program still cannot run and requires extra configuration	- User manual is poorly formatted with some formatting issues - It takes time to understand the user manual - A few screenshots added - By following user manual, the program can successfully run with some extra configuration	**- User manual is formatted well with some formatting issues** **- It will take some time to understand the user manual but acceptable with the provided (a few) screenshots** **- By following user manual, the program can successfully run with no extra configuration required**	- Majorty of user manual is formatted professionally - User manual is in detail and relatively easy to understand with some screenshots - By following user manual, the program can successfully run without any issue	- User manual is formatted in a professional manner - User manual provide very detailed and easy to understand instructions with descriptive screenshots - By following user manual, the program can successfully run without any issue
E. Deductions		- Your file naming is correct (no marks deducted) - contribution declaration clearly mentions the contribution (in percentage) and mentioned majority of the tasks understaken by all the team members (no marks deducted) - same type of application developed as presented in Week 5 (no marks deducted)	**- Your file namings are wrong (1 mark deducted)** **- contribution declaration not made (1 mark deducted)** **- contribution declaration made does not help understand the contributions made by each team member (1 mark deducted)** **- type of application is different than the one presented in Week 5 (- 25% of team marks)**			

Late Penalty / Day	10%	0 Day	1 Day	2 Day	3 Day	4 Day
	1					
Final Total marks	10.27					
Comments:	Software Design 1. The data types of attributes are needed in a detailed class diagram. 2. The boundary class is not only responsible for displaying output but also taking input from the user. So, keep that in mind when designing. 3. The functionalities of the system is quite small so having 1 control class is enough. And making your control classes the child classes of an entity class is not advisable. Implementation 1. You are jamming all your classes into 1 main.py file. This is not advisable. It is a bad design. 2. There is not enough documentation in your code. You are hardcoding the topics and keywords. 3. Please, do not put sarcastic feedback messages to your users. Try to provide helpful feedback with hints about the expected input. 4. Try to follow the client requirements as users can only upload .doc or .pdf files and has to provide file path. User Manual 1. Your user should not do a pull from the git. Please fix your instructions accordingly. 2. You should provide some troubleshooting steps to your user and mention the system environment as well. 3. A table of content can improve the document. Other comments 1. Your submitted class diagram partially matches your code. For example: you do not have a CMS class in your design and the methods in the code of that class are not in the design. 2. Please name your submitted files the way it is prescribed in the assignment details.					