CS-258 G11 GROUP
ONLINE CAMPUS SECURITY SYSTEM

Purpose:

IIT Indore has more than thousand security persons, who are instructed to give duties at different places within the campus. Therefore, the objective is to build an Online Security Management System (Secures) through which the entire security system within the campus can be controlled in an efficient manner. This system will assist the Security manager to control security efficiently as manual calculation/operation is a heavy task for the security manager.

Functional Requirements:

1. Each security member will have their account which they can access by logging in with their registered email and security verification through OTP system. They can update their details as frequently as they want.
2. The manager would have a unique confidential password through which only he can access the system.
3. The system should allow two views-one for the manager, and the other for each security member. Thereafter, the viewable features and functionalities would be different corresponding to the role a person has logged in from.
4. Each security person's account contains all information about them, such as each member would have a unique identification number and all other personal details like phone number, email, age, etc. will be stored in the database.
5. Each place to be monitored will also have a unique identification number to easily handle the process of allocation of duties.
6. Maintain a routine consisting of the date, start time and end time of the duty of each guard and the place monitored by the guard. The entire routine for each day corresponding to each place should be displayed on the home page in a user-friendly tabular layout.
7. Each person can view the routine, that is, at which place each person is allocated the duty after logging in and also their contact details are visible to all of them so that they can do faster communication in case of any mishap.

8. All the places should be covered by at least one security person.
9. Each member would have a fixed number of paid leaves (ex. 3 leaves per month). Exceeding this limit will cause a fixed amount to be deducted from the salary for each extra leave.
10. The members have to send a ‘Leave Request’ to the manager before taking a leave so that we are sure that each place is under the supervision of at least a security person.
11. The manager’s duty is to approve/reject the request. If approved by the manager, only then the leave would be granted.
12. The system should maintain a record of those members who have applied to take up an extra duty. Whenever a leave request is granted to someone, another person from this list will be allotted duty at the vacant place.
13. The system should provide search features based on ‘Name of security member’ as well as the ‘Date of the Month’. The first method would render the personal details along with the work history of the searched person. The latter method shall list who all members are monitoring what all places on that particular date and the start and end times of their duties.
14. At the end of each month, the system should calculate each member’s salary after considering all the leaves taken and extra duties delivered.
15. After the salary calculation, the system shall generate a salary summary for each member and then send it in their mail.
16. Maintain a record of mishappening/incidents that occurred previously in any of the monitored places and display them.

Non-functional requirements:

1. Readability/Maintainability - A good naming strategy, use of proper comments, clear control flow, indentation and spacing makes our code more readable, maintainable as we can fix our bugs more easily as compared to a messy code.
2. Usability - As it is more readable it will increase its usability and also people will find it easier to contribute if it is open-source.
3. Modularity/Reusability - Making separate functions for each task. It makes our code more modular, reusable as we can just call it at the place we require without writing the whole code again and again.
4. Flexibility - If we want to fix any bug or modify our system code it is much easier because we just have to change at very few places only. Our code is much more flexible and modifiable.
5. Testability- As we have separate functions for each task, it's very easy to test them individually.
6. Reliability- Maintain a failure-free system by proper management and calculation of data. Data calculation should be done only for the concerned person. It makes our code testable and reliable.
7. Security - The signup process is secured by the OTP system that will verify the email address of the person, along with password verification.
8. Exception handling - If during signup or login the user leaves any of the required fields empty then the system will not allow the user to login and will show the message to fill all the fields first and then login. And also if the OTP entered by the user during email verification is incorrect then also the alert will be shown.
9. Efficiency - The software should utilize the optimal amount of resources, without any wastage of memory or processor. The salary calculation meets this requirement by making accurate calculations, and hence is efficient, as we have not used any extra resource than needed.
10. Adaptability- The system should be able to get adapted reasonably as the user requirements alter.
11. Scalability- The system should work efficiently even if the number of users increases. The database of the system can handle larger amounts of data as well.
12. In-time – The algorithms/calculations don't take much time to be built up.
13. Portability – The system defined above can be executed on various types of operating systems without converting the program to other

languages, and with little or no modifications. The system uses standard softwares and technologies, thus can be efficiently used cross-platform.

14. Interoperability – Two or more functional units can process data cooperatively, for our system. Various codes are working parallely.

15. Response Time - The site should load in 2-3 seconds every time the page is refreshed or new data is processed. The system should be quick enough to avoid data redundancies or data mismatches and inconsistencies.