Title: Crime Management System

Team members:

Shreya Varma: PES1UG20CS403 Shria Guntunur: PES1UG20CS411 Shruthi Pai: PES1UG20CS416

Shubangi Saxena: PES1UG20CS418

Section: 'G'

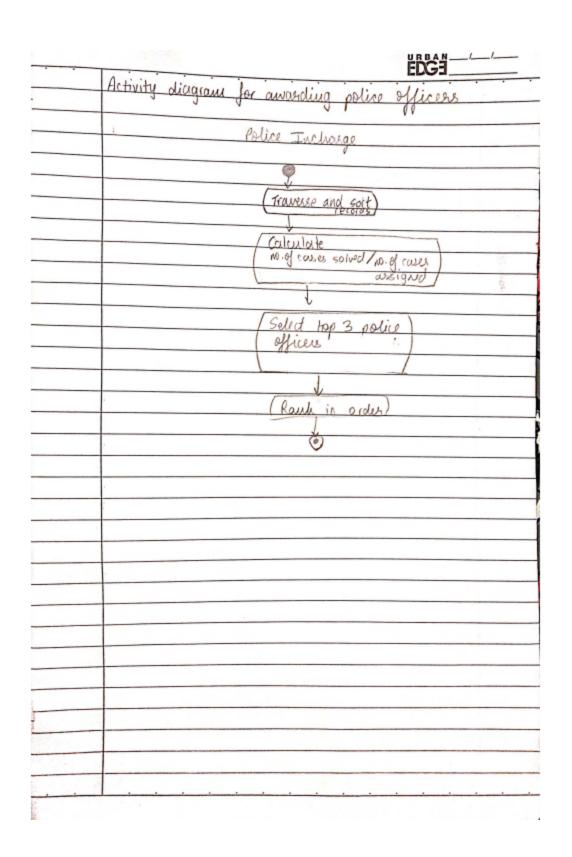
Synopsis:

In this project, we are aiming to create a crime management system. This platform can be used by 4 sets of users- the civilians, the police personnel (Police officers and Police incharges) and head officers. The civilian can register themselves on this site, lodge a complaint as well as see their complaint history along with the status of each crime reported by them. The police personnel further consist of police officers and the incharge officers. There is one police station in a given location and every police station has 1 incharge and multiple police officers. Once a complaint is lodged by a civilian, the incharge views it(only if it has been registered in that particular police station) and assigns it to one of his/her police officers based on their specializations (murder, robbery, kidnapping etc.). The police incharge is also responsible for creating a ranking list to award the top 3 police officers. Once a case has been assigned to a police officer, he/she can view it and make the required updates to the case. The police incharge can view all these updates. The head officer is incharge of managing the police incharges' records and the police incharge is responsible for managing the police officers' records. The head officer can also view inference of crime rates based on the location of the crime committed. Overall, our project aims to create a centralized method of maintaining crime records.

Use case diagram:

	Use Case Diagram.
	Q (File FIR)
	Civilian View approved updates of case)
	Login (indude) (Regisles)
	update investigation report) offices
	Award Fredrice
	(Report to incharge) cases
	Tricharee Assign Case to Police Offices (and >
	(Add closed are to DB)
	View updates of 62e) (extends> (Manage Police Offices) < (Add police Head off)
C.P.	olling 1/1-1
	had Folice inchange Tricharges
	Crime Suspects. (Inference on evine rates in different awas)

Ç	ass Diagra	М		
	0.1:	Officer	-	Police Tudias
Civilian	rona	Office		· Naw: Shing
+ Name . Shing	+ Now	: String		+ DOB : Date
+206; Date		+ DOB: Dale		10; Tut.
+ email: Shing	101	INT	-	Email: Smine
+ Addres: Shirg		+ emailistring		Addies : String
+ PKNO: Jut		THURB: SMIG		+ PhNo: Int
+ login()	4 (NNo	+ PhNo; Int - - login()		- login()
+ file-fir()	- logi	۸()	Α.	- asign-cosel
+ view-up()	- PP	ort()		add-cc-to
1	50010)	(2000)		riew-ir()
		14	A -	aword()
		1	7	- manage-po
his.	CS	IR	-	1*
	cr-cs()	tup ic		Wou
		MOLPSON		Wou
*	111	1,.*		₩1 1 201°
FIR.	Cas	e	. ,	Read Office
+ (D:11)+	-1d:	-ld:int		- Nome: Shi
+ Now: Shing	HOM	thom: Shing		- DoB: Date
+ Dozaiphion: Shing	+ 140	18: Shing		ID : Int
+Type ; Shing	+De	sc: String	.~	-Email: Sta
t search ()	t 0ρ-	d: boolean	N .	-Adres: S
1.	- d	bse()		- PhNo: I
		earch()		- manage-
			R	- dvalyze_



Activity Diagram for Police	Offices:
Police Incharge	Police Office
0	00
V	
(theck (alegory of	
Case Received	W
(Solort 1: 11:	
Treate de l'in	
Select police officer records for this category	
334	
Order in increasing	
order of number	
of active cases	
(C1.11)	
Swift the police	
Select the police offices with the first rank	
ANS TABLE	
	Assign case to
	Police Offices
	/
	(e)

Civilian	Headsfile
	W.
0	
(5) 50	
File FIR at	
a local police	
Station	N.
	Views FIR
	with the location
	[Add to a con]
	Add FIR to existing
	Complaint database
	Count NO. 01 FIR'S
	per location
	1
	· Display the location
	with the highest
	aim reti
	(6)

Design principles:

- **1. High Cohesion:** MVC enables logical grouping of related actions on a controller together in a controller class.
- 2. Information Expert: The HeadOfficer class contains all the information about the PoliceIncharge class and can manage it. Similarly, the Police Incharge class manages the Police Officer class.
- **3. Controller:** MVC provides a controller class which is an interface between the model and the view and can transfer information between the two classes.

Design Patterns:

- 1. **Singleton:** Singleton pattern involves a single class responsible for creating an object and ensuring that only a single instance ever gets created. SpringBoot framework implements Singleton pattern by default.
- 2. Proxy: When a method is annotated with @Transactional, Spring dynamically creates a proxy class that implements the same interfaces as the class we annotate. When clients make calls to the class, the calls are intercepted using the proxy. It is used so that the class is protected in case of a roll back.
- Design Access Object: Controller communicates with the service, and the service communicates with the Repository. The service is in the dark about the repository functionality.