Project:
HealthPlus

Scope:
Project 'HealthPlus'

Profile:
Default

Results:
Enabled coding rules: 30

- Checkstyle: 30

Problems found: 1433

- Checkstyle: 1433

Time statistics:

- Started: Mon Apr 08 17:02:55 EDT 2024
  - Initialization: 00:00:00.012
  - Checkstyle: 00:00:01.713
  - Gathering results: 00:00:00.654
- Finished: Mon Apr 08 17:02:58 EDT 2024

Detailed Results:
HealthPlus

- Efficiency
  - Hide Utility Class Constructor
    - Validate
      - Utility classes should not have a public or default constructor. - line: 4
- Maintainability
  - Anon Inner Length
    - AdminController
      - Anonymous inner class length is 31 lines (max allowed is 20). - line: 1217
    - DatabaseOperator
      - Anonymous inner class length is 38 lines (max allowed is 20). - line: 52
    - ReceptionistController
      - Anonymous inner class length is 42 lines (max allowed is 20). - line: 1067
      - Anonymous inner class length is 35 lines (max allowed is 20). - line: 1072
  - Boolean Expression Complexity
    - AdminController
      - Boolean expression complexity is 4 (max allowed is 3). - line: 842
  - Cyclomatic Complexity
    - Admin
      - Cyclomatic Complexity is 28 (max allowed is 10). - line: 153
    - AdminController
      - Cyclomatic Complexity is 21 (max allowed is 10). - line: 777
    - CashierController
      - Cyclomatic Complexity is 14 (max allowed is 10). - line: 306
    - Doctor
      - Cyclomatic Complexity is 12 (max allowed is 10). - line: 366
    - DoctorController
      - Cyclomatic Complexity is 17 (max allowed is 10). - line: 605

        - Cyclomatic Complexity is 15 (max allowed is 10). - line: 1034
    - LabAssistantController
        - Cyclomatic Complexity is 22 (max allowed is 10). - line: 228
    - LabReportPreviewController
        - Cyclomatic Complexity is 17 (max allowed is 10). - line: 42
    - Pharmacist
        - Cyclomatic Complexity is 13 (max allowed is 10). - line: 261
        - Cyclomatic Complexity is 11 (max allowed is 10). - line: 975
    - Receptionist
        - Cyclomatic Complexity is 12 (max allowed is 10). - line: 180
        - Cyclomatic Complexity is 18 (max allowed is 10). - line: 335
        - Cyclomatic Complexity is 17 (max allowed is 10). - line: 513
    - ReceptionistController
        - Cyclomatic Complexity is 20 (max allowed is 10). - line: 485
    - ReportsController
        - Cyclomatic Complexity is 16 (max allowed is 10). - line: 206
        - Cyclomatic Complexity is 16 (max allowed is 10). - line: 360
        - Cyclomatic Complexity is 22 (max allowed is 10). - line: 786
    - UserAccountController
        - Cyclomatic Complexity is 11 (max allowed is 10). - line: 96
- Empty Statement
    - ReceptionistController
        - Empty statement. - line: 1616
- Redundant Modifier
    - NewUserController
        - Redundant 'public' modifier. - line: 23
    - UserOptionPopOverController
        - Redundant 'public' modifier. - line: 25
- Simplify Boolean Expression
    - AddNewDrugController
        - Expression can be simplified. - line: 250
    - AdminController
        - Expression can be simplified. - line: 811
        - Expression can be simplified. - line: 845
        - Expression can be simplified. - line: 856
        - Expression can be simplified. - line: 856
        - Expression can be simplified. - line: 987
        - Expression can be simplified. - line: 1516
        - Expression can be simplified. - line: 1545
        - Expression can be simplified. - line: 1572
        - Expression can be simplified. - line: 1584
    - Cashier
        - Expression can be simplified. - line: 143
    - CashierController
        - Expression can be simplified. - line: 879
        - Expression can be simplified. - line: 908
        - Expression can be simplified. - line: 935
        - Expression can be simplified. - line: 947
    - Doctor
        - Expression can be simplified. - line: 407
        - Expression can be simplified. - line: 420
    - DoctorController
        - Expression can be simplified. - line: 590
        - Expression can be simplified. - line: 1104

        - Expression can be simplified. - line: 1104
        - Expression can be simplified. - line: 1120
        - Expression can be simplified. - line: 1364
        - Expression can be simplified. - line: 1598
        - Expression can be simplified. - line: 1624
        - Expression can be simplified. - line: 1651
        - Expression can be simplified. - line: 1663
        - Expression can be simplified. - line: 1744
    - LabAssistantController
        - Expression can be simplified. - line: 1340
        - Expression can be simplified. - line: 1393
        - Expression can be simplified. - line: 1419
        - Expression can be simplified. - line: 1465
        - Expression can be simplified. - line: 1477
    - NewDoctorTimeSlotController
        - Expression can be simplified. - line: 118
    - NewMessageController
        - Expression can be simplified. - line: 136
    - PharmacistController
        - Expression can be simplified. - line: 333
        - Expression can be simplified. - line: 459
        - Expression can be simplified. - line: 994
        - Expression can be simplified. - line: 1047
        - Expression can be simplified. - line: 1073
        - Expression can be simplified. - line: 1100
        - Expression can be simplified. - line: 1112
    - PopupAskController
        - Expression can be simplified. - line: 62
    - ReadMessageController
        - Expression can be simplified. - line: 75
    - Receptionist
        - Expression can be simplified. - line: 413
        - Expression can be simplified. - line: 1124
        - Expression can be simplified. - line: 1283
    - ReceptionistController
        - Expression can be simplified. - line: 567
        - Expression can be simplified. - line: 610
        - Expression can be simplified. - line: 1034
        - Expression can be simplified. - line: 1039
        - Expression can be simplified. - line: 1265
        - Expression can be simplified. - line: 1294
        - Expression can be simplified. - line: 1321
        - Expression can be simplified. - line: 1333
    - SettingsController
        - Expression can be simplified. - line: 157
    - SysUserController
        - Expression can be simplified. - line: 155
    - Validate
        - Expression can be simplified. - line: 103
        - Expression can be simplified. - line: 159
        - Expression can be simplified. - line: 171
- Unused Imports
    - AdminController.java
        - Unused import - javafx.scene.control.Tab. - line: 53

- AdminController.java
  - Unused import - Doctor.DoctorController. - line: 2
- AdminController.java
  - Unused import - javafx.stage.Modality. - line: 39
- AdminController.java
  - Unused import - javafx.scene.control.SingleSelectionModel. - line: 52
- AllAppointmentsController.java
  - Unused import - javafx.stage.Screen. - line: 14
- AllAppointmentsController.java
  - Unused import - javafx.collections.ObservableList. - line: 9
- AllAppointmentsController.java
  - Unused import - javafx.stage.StageStyle. - line: 16
- AllAppointmentsController.java
  - Unused import - javafx.scene.Scene. - line: 12
- AllAppointmentsController.java
  - Unused import - javafx.geometry.Rectangle2D. - line: 11
- AllMessagesController.java
  - Unused import - javafx.scene.layout.VBox. - line: 17
- AllMessagesController.java
  - Unused import - javafx.geometry.Pos. - line: 9
- AllMessagesController.java
  - Unused import - javafx.scene.layout.Priority. - line: 16
- CashierController.java
  - Unused import - javafx.scene.input.MouseEvent. - line: 42
- CurrentUserSummaryController.java
  - Unused import - javafx.collections.ObservableList. - line: 8
- CurrentUserSummaryController.java
  - Unused import - javafx.stage.Screen. - line: 15
- CurrentUserSummaryController.java
  - Unused import - javafx.scene.Scene. - line: 11
- CurrentUserSummaryController.java
  - Unused import - javafx.geometry.Rectangle2D. - line: 10
- CurrentUserSummaryController.java
  - Unused import - javafx.stage.StageStyle. - line: 17
- CurrentUserSummaryController.java
  - Unused import - org.controlsfx.control.textfield.TextFields. - line: 18
- Doctor.java
  - Unused import - java.time.LocalDate. - line: 6
- LabAssistant.java
  - Unused import - java.util.Arrays. - line: 10
- LabReportPreviewController.java
  - Unused import - java.util.HashMap. - line: 8
- LabReportPreviewController.java
  - Unused import - java.time.LocalDate. - line: 4
- LogoutController.java
  - Unused import - Cashier.CashierController. - line: 3
- LogoutController.java
  - Unused import - javafx.scene.control.PasswordField. - line: 19
- LogoutController.java
  - Unused import - Receptionist.ReceptionistController. - line: 8
- LogoutController.java
  - Unused import - javafx.scene.Node. - line: 16
- LogoutController.java
  - Unused import - Pharmacist.PharmacistController. - line: 7

- LogoutController.java
  - Unused import - javafx.scene.control.TextField. - line: 20
- LogoutController.java
  - Unused import - Doctor.DoctorController. - line: 5
- LogoutController.java
  - Unused import - java.sql.SQLException. - line: 10
- LogoutController.java
  - Unused import - javafx.stage.StageStyle. - line: 24
- LogoutController.java
  - Unused import - Admin.AdminController. - line: 2
- LogoutController.java
  - Unused import - java.util.ArrayList. - line: 11
- MainApp.java
  - Unused import - Cashier.CashierController. - line: 3
- MainApp.java
  - Unused import - javafx.fxml.FXMLLoader. - line: 12
- MainApp.java
  - Unused import - javafx.scene.Parent. - line: 14
- MainApp.java
  - Unused import - Receptionist.ReceptionistController. - line: 8
- MainApp.java
  - Unused import - Doctor.DoctorController. - line: 5
- MainApp.java
  - Unused import - LabAssistant.LabAssistantController. - line: 6
- MainApp.java
  - Unused import - javafx.scene.layout.Pane. - line: 16
- MainApp.java
  - Unused import - Pharmacist.PharmacistController. - line: 7
- MainApp.java
  - Unused import - java.io.IOException. - line: 9
- MainApp.java
  - Unused import - Admin.AdminController. - line: 2
- NewDoctorTimeSlotController.java
  - Unused import - javafx.util.converter.LocalTimeStringConverter. - line: 18
- NewDoctorTimeSlotController.java
  - Unused import - java.time.format.FormatStyle. - line: 9
- NewDoctorTimeSlotController.java
  - Unused import - java.time.LocalTime. - line: 7
- NewDoctorTimeSlotController.java
  - Unused import - java.time.format.DateTimeFormatter. - line: 8
- NewMessageController.java
  - Unused import - javafx.collections.ObservableList. - line: 8
- PharmacistController.java
  - Unused import - javafx.scene.paint.Color. - line: 49
- PharmacistController.java
  - Unused import - javafx.scene.input.MouseEvent. - line: 46
- PopupAskController.java
  - Unused import - java.util.ArrayList. - line: 9
- PopupAskController.java
  - Unused import - javafx.scene.image.ImageView. - line: 17
- PopupAskController.java
  - Unused import - javafx.collections.FXCollections. - line: 10
- PopupAskController.java
  - Unused import - javafx.stage.Screen. - line: 21

- PopupAskController.java
  - Unused import - javafx.scene.Scene. - line: 14
- PopupAskController.java
  - Unused import - javafx.collections.ObservableList. - line: 11
- PopupAskController.java
  - Unused import - javafx.scene.image.Image. - line: 16
- PopupAskController.java
  - Unused import - javafx.scene.layout.VBox. - line: 20
- PopupAskController.java
  - Unused import - javafx.geometry.Rectangle2D. - line: 13
- PopupAskController.java
  - Unused import - javafx.geometry.Pos. - line: 12
- PopupAskController.java
  - Unused import - javafx.scene.control.cell.PropertyValueFactory. - line: 15
- PopupAskController.java
  - Unused import - javafx.stage.StageStyle. - line: 23
- PopupAskController.java
  - Unused import - javafx.scene.layout.Priority. - line: 19
- PrescriptionListController.java
  - Unused import - javafx.scene.Node. - line: 10
- PrescriptionListController.java
  - Unused import - javafx.scene.input.KeyEvent. - line: 13
- PrescriptionListController.java
  - Unused import - javafx.scene.input.KeyCode. - line: 11
- ReadMessageController.java
  - Unused import - javafx.scene.image.ImageView. - line: 15
- ReadMessageController.java
  - Unused import - javafx.scene.control.cell.PropertyValueFactory. - line: 13
- ReadMessageController.java
  - Unused import - javafx.collections.ObservableList. - line: 8
- ReadMessageController.java
  - Unused import - java.util.ArrayList. - line: 6
- ReadMessageController.java
  - Unused import - javafx.scene.layout.Priority. - line: 17
- ReadMessageController.java
  - Unused import - javafx.scene.image.Image. - line: 14
- ReadMessageController.java
  - Unused import - javafx.collections.FXCollections. - line: 7
- ReadMessageController.java
  - Unused import - javafx.scene.layout.VBox. - line: 18
- ReadMessageController.java
  - Unused import - javafx.geometry.Pos. - line: 10
- Receptionist.java
  - Unused import - java.time.LocalDateTime. - line: 6
- Receptionist.java
  - Unused import - java.time.format.DateTimeFormatter. - line: 7
- RefundController.java
  - Unused import - javafx.scene.control.Button. - line: 12
- RefundController.java
  - Unused import - javafx.event.ActionEvent. - line: 7
- ReportsController.java
  - Unused import - javafx.scene.input.MouseEvent. - line: 29
- ReportsController.java
  - Unused import - javafx.scene.paint.Color. - line: 31

- ReportsController.java
  - Unused import - javafx.event.EventHandler. - line: 19
- SettingsController.java
  - Unused import - java.io.FileInputStream. - line: 7
- SysUserController.java
  - Unused import - org.controlsfx.control.textfield.TextFields. - line: 21
- SysUserController.java
  - Unused import - javafx.collections.ObservableList. - line: 13
- SysUserController.java
  - Unused import - java.util.HashMap. - line: 12
- SysUserController.java
  - Unused import - java.util.Date. - line: 11
- SysUserController.java
  - Unused import - java.text.SimpleDateFormat. - line: 7
- UserAccount.java
  - Unused import - javafx.scene.control.Button. - line: 8
- UserOptionPopOverController.java
  - Unused import - javafx.scene.control.Button. - line: 15
- UserOptionPopOverController.java
  - Unused import - javafx.event.ActionEvent. - line: 8
- UserOptionPopOverController.java
  - Unused import - javafx.scene.Parent. - line: 11
- UserOptionPopOverController.java
  - Unused import - javafx.stage.Stage. - line: 17

- Visibility Modifier
  - AdminController
    - Variable 'admin' must be private and have accessor methods. - line: 75
    - Variable 'username' must be private and have accessor methods. - line: 80
    - Variable 'userLog' must be private and have accessor methods. - line: 111
    - Variable 'userType' must be private and have accessor methods. - line: 382
    - Variable 'chooser' must be private and have accessor methods. - line: 940
    - Variable 'log' must be private and have accessor methods. - line: 1002
    - Variable 'suspendList' must be private and have accessor methods. - line: 1169
    - Variable 'profileImage' must be private and have accessor methods. - line: 1274
    - Variable 'editProfilePicButton' must be private and have accessor methods. - line: 1276
    - Variable 'chooser2' must be private and have accessor methods. - line: 1277
    - Variable 'path' must be private and have accessor methods. - line: 1279
    - Variable 'name' must be private and have accessor methods. - line: 1280
  - AdminMessageController
    - Variable 'receiver' must be private and have accessor methods. - line: 30
    - Variable 'message' must be private and have accessor methods. - line: 31
  - AllAppointmentsController
    - Variable 'log' must be private and have accessor methods. - line: 142
    - Variable 'log2' must be private and have accessor methods. - line: 143
    - Variable 'docNames' must be private and have accessor methods. - line: 144
  - AllMessagesController
    - Variable 'messagesTable' must be private and have accessor methods. - line: 44
  - Appointment
    - Variable 'startTime' must be private and have accessor methods. - line: 9
    - Variable 'endTime' must be private and have accessor methods. - line: 10
  - AppointmentSuccessController
    - Variable 'saveSuccess' must be private and have accessor methods. - line: 42
  - BillPreviewController
    - Variable 'saveSuccess' must be private and have accessor methods. - line: 27

- CashierController
  - Variable 'cashier' must be private and have accessor methods. - line: 55
  - Variable 'username' must be private and have accessor methods. - line: 60
  - Variable 'patientLog' must be private and have accessor methods. - line: 258
  - Variable 'profileImage' must be private and have accessor methods. - line: 615
  - Variable 'editProfilePicButton' must be private and have accessor methods. - line: 617
  - Variable 'chooser' must be private and have accessor methods. - line: 618
  - Variable 'path' must be private and have accessor methods. - line: 620
  - Variable 'name' must be private and have accessor methods. - line: 621
- DatabaseOperator
  - Variable 'dbClassName' must be private and have accessor methods. - line: 96
  - Variable 'CONNECTION' must be private and have accessor methods. - line: 97
  - Variable 'c' must be private and have accessor methods. - line: 99
- Doctor
  - Variable 'slmcRegNo' must be private and have accessor methods. - line: 41
- DoctorController
  - Variable 'doc' must be private and have accessor methods. - line: 59
  - Variable 'username' must be private and have accessor methods. - line: 60
  - Variable 'allergyView' must be private and have accessor methods. - line: 484
  - Variable 'patientLog' must be private and have accessor methods. - line: 801
  - Variable 'popup' must be private and have accessor methods. - line: 848
  - Variable 'saveProgress' must be private and have accessor methods. - line: 1370
  - Variable 'profileImage' must be private and have accessor methods. - line: 1442
  - Variable 'editProfilePicButton' must be private and have accessor methods. - line: 1444
  - Variable 'chooser' must be private and have accessor methods. - line: 1445
  - Variable 'path' must be private and have accessor methods. - line: 1447
  - Variable 'name' must be private and have accessor methods. - line: 1448
- Drug
  - Variable 'amount' must be private and have accessor methods. - line: 12
- ErrorController
  - Variable 'formatLabel' must be private and have accessor methods. - line: 33
- LabAssistantController
  - Variable 'lab' must be private and have accessor methods. - line: 63
  - Variable 'username' must be private and have accessor methods. - line: 64
  - Variable 'fxmlLoader' must be private and have accessor methods. - line: 65
  - Variable 'serachType' must be private and have accessor methods. - line: 123
  - Variable 'appointmentIDtext' must be private and have accessor methods. - line: 125
  - Variable 'patientNametext' must be private and have accessor methods. - line: 126
  - Variable 'patientAgetext' must be private and have accessor methods. - line: 127
  - Variable 'patientGendertext' must be private and have accessor methods. - line: 128
  - Variable 'patientConsultanttext' must be private and have accessor methods. - line: 129
  - Variable 'reportTabs' must be private and have accessor methods. - line: 131
  - Variable 'pt' must be private and have accessor methods. - line: 133
  - Variable 'lpt' must be private and have accessor methods. - line: 134
  - Variable 'bg' must be private and have accessor methods. - line: 135
  - Variable 'cbc' must be private and have accessor methods. - line: 136
  - Variable 'lft' must be private and have accessor methods. - line: 137
  - Variable 'rft' must be private and have accessor methods. - line: 138
  - Variable 'cpk' must be private and have accessor methods. - line: 139
  - Variable 'hiv' must be private and have accessor methods. - line: 140
  - Variable 'precrptionDate' must be private and have accessor methods. - line: 437
  - Variable 'testList' must be private and have accessor methods. - line: 438
  - Variable 'labAppointmentTable' must be private and have accessor methods. - line: 514
  - Variable 'profileImage' must be private and have accessor methods. - line: 1082

- Variable 'editProfilePicButton' must be private and have accessor methods. - line: 1084
- Variable 'chooser' must be private and have accessor methods. - line: 1085
- Variable 'path' must be private and have accessor methods. - line: 1087
- Variable 'name' must be private and have accessor methods. - line: 1088
- Variable 'popOver' must be private and have accessor methods. - line: 1156

- LabReportPreviewController
  - Variable 'report' must be private and have accessor methods. - line: 40
- LoginController
  - Variable 'popOver' must be private and have accessor methods. - line: 256
- NewMessageController
  - Variable 'receivertxt' must be private and have accessor methods. - line: 53
  - Variable 'userid' must be private and have accessor methods. - line: 78
- PatientAccountSuccessController
  - Variable 'saveSuccess' must be private and have accessor methods. - line: 42
- PharmacistController
  - Variable 'pharmacist' must be private and have accessor methods. - line: 61
  - Variable 'username' must be private and have accessor methods. - line: 66
  - Variable 'view' must be private and have accessor methods. - line: 196
  - Variable 'profileImage' must be private and have accessor methods. - line: 718
  - Variable 'editProfilePicButton' must be private and have accessor methods. - line: 720
  - Variable 'chooser' must be private and have accessor methods. - line: 721
  - Variable 'path' must be private and have accessor methods. - line: 723
  - Variable 'name' must be private and have accessor methods. - line: 724
- PopoverController
  - Variable 'text' must be private and have accessor methods. - line: 22
- PrescriptionListController
  - Variable 'lab' must be private and have accessor methods. - line: 33
  - Variable 'prescList' must be private and have accessor methods. - line: 34
- ReceptionistController
  - Variable 'receptionist' must be private and have accessor methods. - line: 53
  - Variable 'username' must be private and have accessor methods. - line: 54
  - Variable 'newPatient' must be private and have accessor methods. - line: 467
  - Variable 'reduceQueButton' must be private and have accessor methods. - line: 468
  - Variable 'queLabel' must be private and have accessor methods. - line: 469
  - Variable 'patientLog' must be private and have accessor methods. - line: 625
  - Variable 'days' must be private and have accessor methods. - line: 778
  - Variable 'appCancelClearButton' must be private and have accessor methods. - line: 1054
  - Variable 'appDatePicker' must be private and have accessor methods. - line: 1066
  - Variable 'dayCellFactory' must be private and have accessor methods. - line: 1067
  - Variable 'profileImage' must be private and have accessor methods. - line: 1386
  - Variable 'editProfilePicButton' must be private and have accessor methods. - line: 1388
  - Variable 'chooser' must be private and have accessor methods. - line: 1389
  - Variable 'path' must be private and have accessor methods. - line: 1391
  - Variable 'name' must be private and have accessor methods. - line: 1392
- RefundController
  - Variable 'cashier' must be private and have accessor methods. - line: 25
  - Variable 'closeRefund' must be private and have accessor methods. - line: 115
- ReportsController
  - Variable 'patientAttendenceCombo' must be private and have accessor methods. - line: 59
  - Variable 'patientAttendence' must be private and have accessor methods. - line: 60
- SettingsController
  - Variable 'result' must be private and have accessor methods. - line: 53

        - Variable 'inputStream' must be private and have accessor methods. - line: 54
        - Variable 'chooser' must be private and have accessor methods. - line: 134
    - SuccessIndicatorController
        - Variable 'saveSuccess' must be private and have accessor methods. - line: 27
    - User
        - Variable 'dbOperator' must be private and have accessor methods. - line: 30
        - Variable 'username' must be private and have accessor methods. - line: 31
        - Variable 'userID' must be private and have accessor methods. - line: 32
        - Variable 'userType' must be private and have accessor methods. - line: 33
        - Variable 'database' must be private and have accessor methods. - line: 35
        - Variable 'dbUsername' must be private and have accessor methods. - line: 36
        - Variable 'dbPassword' must be private and have accessor methods. - line: 37
    - UserAccountController
        - Variable 'admin' must be private and have accessor methods. - line: 26
        - Variable 'usertype' must be private and have accessor methods. - line: 27
    - WarningController
        - Variable 'formatLabel' must be private and have accessor methods. - line: 33
- Reliability
    - Design For Extension
        - AddNewDrugController
            - Class 'AddNewDrugController' looks like designed for extension (can be subclassed), but the method 'loadGenericNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AddNewDrugController' final or making the method 'loadGenericNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 59
            - Class 'AddNewDrugController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AddNewDrugController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 290
        - Admin
            - Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getOnlineInfo2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getOnlineInfo2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 424
            - Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSysUserCount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSysUserCount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 439
            - Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getPatientCount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getPatientCount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 454
            - Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getAllPatientCount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getAllPatientCount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 469
            - Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getOnlineCount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the

method 'getOnlineCount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 484

- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'checkConnection' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'checkConnection' static/final/abstract/empty, or adding allowed annotation for the method. - line: 500
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'export' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'export' static/final/abstract/empty, or adding allowed annotation for the method. - line: 515
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSchemaSize' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSchemaSize' static/final/abstract/empty, or adding allowed annotation for the method. - line: 558
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getPatientAttendence' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getPatientAttendence' static/final/abstract/empty, or adding allowed annotation for the method. - line: 607
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getDoctorNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getDoctorNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 635
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'lastMonthsReports' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'lastMonthsReports' static/final/abstract/empty, or adding allowed annotation for the method. - line: 650
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getDocAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getDocAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 685
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getLabAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getLabAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 705
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getCancelledDocAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getCancelledDocAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 725
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getCancelledLabAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getCancelledLabAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 745
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getStockSummary' does not have javadoc that explains how to do that safely. If class is

not designed for extension consider making the class 'Admin' final or making the method 'getStockSummary' static/final/abstract/empty, or adding allowed annotation for the method. - line: 765

- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getDrugGenericInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getDrugGenericInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 818
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getDrugNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getDrugNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 846
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getDrugAmounts' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getDrugAmounts' static/final/abstract/empty, or adding allowed annotation for the method. - line: 860
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSupplierNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSupplierNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 875
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSupplierSummary' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSupplierSummary' static/final/abstract/empty, or adding allowed annotation for the method. - line: 901
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'lastTotalIncome' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'lastTotalIncome' static/final/abstract/empty, or adding allowed annotation for the method. - line: 942
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'pharmacyIncome' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'pharmacyIncome' static/final/abstract/empty, or adding allowed annotation for the method. - line: 969
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'laboratoryIncome' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'laboratoryIncome' static/final/abstract/empty, or adding allowed annotation for the method. - line: 996
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'appointmentIncome' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'appointmentIncome' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1023
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSysUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSysUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1051

- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'suspendUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'suspendUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1072
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'unsuspendUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'unsuspendUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1090
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getsuspendUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getsuspendUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1108
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getActiveUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getActiveUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1129
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'getSuspendedUsers' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'getSuspendedUsers' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1151
- Class 'Admin' looks like designed for extension (can be subclassed), but the method 'resetPassword' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Admin' final or making the method 'resetPassword' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1173

- AdminController
  - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'filldatabaseStorageChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'filldatabaseStorageChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 113
  - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'loadDatabaseInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'loadDatabaseInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 919
  - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'loadUsers' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'loadUsers' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1004
  - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'fillOnlineUsers' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'fillOnlineUsers' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1089
  - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'fillStorageInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or

making the method 'fillStorageInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1113
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'fillAccountCounts' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'fillAccountCounts' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1143
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'getSuspendedInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'getSuspendedInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1212
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1282
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1331
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1592
    - Class 'AdminController' looks like designed for extension (can be subclassed), but the method 'addFocusListener' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminController' final or making the method 'addFocusListener' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1859
- AdminMessageController
    - Class 'AdminMessageController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AdminMessageController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
- AllAppointmentsController
    - Class 'AllAppointmentsController' looks like designed for extension (can be subclassed), but the method 'loadDoctorNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllAppointmentsController' final or making the method 'loadDoctorNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 146
    - Class 'AllAppointmentsController' looks like designed for extension (can be subclassed), but the method 'load' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllAppointmentsController' final or making the method 'load' static/final/abstract/empty, or adding allowed annotation for the method. - line: 167
- AllMessages
    - Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the

method 'setImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34

- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 39
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setImage2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setImage2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 44
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getImage2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getImage2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 49
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setSring' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setSring' static/final/abstract/empty, or adding allowed annotation for the method. - line: 54
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getString' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getString' static/final/abstract/empty, or adding allowed annotation for the method. - line: 59
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setSender' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setSender' static/final/abstract/empty, or adding allowed annotation for the method. - line: 64
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getSender' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getSender' static/final/abstract/empty, or adding allowed annotation for the method. - line: 69
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 74
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 79
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 84
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getDate' does not have javadoc that explains how to do that safely. If class is

not designed for extension consider making the class 'AllMessages' final or making the method 'getDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 89
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setSubject' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setSubject' static/final/abstract/empty, or adding allowed annotation for the method. - line: 94
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getSubject' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getSubject' static/final/abstract/empty, or adding allowed annotation for the method. - line: 99
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setType' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setType' static/final/abstract/empty, or adding allowed annotation for the method. - line: 104
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getType' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getType' static/final/abstract/empty, or adding allowed annotation for the method. - line: 109
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 114
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 119
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'setID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'setID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 124
- Class 'AllMessages' looks like designed for extension (can be subclassed), but the method 'getID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessages' final or making the method 'getID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 129

- AllMessagesController
  - Class 'AllMessagesController' looks like designed for extension (can be subclassed), but the method 'loadMessages' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AllMessagesController' final or making the method 'loadMessages' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
- Appointment
  - Class 'Appointment' looks like designed for extension (can be subclassed), but the method 'getStartTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Appointment' final or making

the method 'getStartTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 17
    - Class 'Appointment' looks like designed for extension (can be subclassed), but the method 'setStartTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Appointment' final or making the method 'setStartTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 21
    - Class 'Appointment' looks like designed for extension (can be subclassed), but the method 'getEndTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Appointment' final or making the method 'getEndTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 25
    - Class 'Appointment' looks like designed for extension (can be subclassed), but the method 'setEndTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Appointment' final or making the method 'setEndTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 29
- AppointmentSuccessController
    - Class 'AppointmentSuccessController' looks like designed for extension (can be subclassed), but the method 'fillAppointmentData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AppointmentSuccessController' final or making the method 'fillAppointmentData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 31
    - Class 'AppointmentSuccessController' looks like designed for extension (can be subclassed), but the method 'saveSuccessExit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'AppointmentSuccessController' final or making the method 'saveSuccessExit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 44
- Availability
    - Class 'Availability' looks like designed for extension (can be subclassed), but the method 'getDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'getDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 22
    - Class 'Availability' looks like designed for extension (can be subclassed), but the method 'setDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'setDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 26
    - Class 'Availability' looks like designed for extension (can be subclassed), but the method 'getTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'getTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 30
    - Class 'Availability' looks like designed for extension (can be subclassed), but the method 'setTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'setTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
    - Class 'Availability' looks like designed for extension (can be subclassed), but the method 'getId' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'getId' static/final/abstract/empty, or adding allowed annotation for the method. - line: 38

- Class 'Availability' looks like designed for extension (can be subclassed), but the method 'setId' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Availability' final or making the method 'setId' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42

- Bill
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getPatientID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getPatientID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 41
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setPatientID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setPatientID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 45
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 49
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 53
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getDoctor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getDoctor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 57
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setDoctor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setDoctor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 61
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getHospital' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getHospital' static/final/abstract/empty, or adding allowed annotation for the method. - line: 65
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setHospital' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setHospital' static/final/abstract/empty, or adding allowed annotation for the method. - line: 69
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getPharmacy' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getPharmacy' static/final/abstract/empty, or adding allowed annotation for the method. - line: 73
  - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setPharmacy' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method

'setPharmacy' static/final/abstract/empty, or adding allowed annotation for the method. - line: 77
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getLaboratory' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getLaboratory' static/final/abstract/empty, or adding allowed annotation for the method. - line: 81
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setLaboratory' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setLaboratory' static/final/abstract/empty, or adding allowed annotation for the method. - line: 85
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getAppointment' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getAppointment' static/final/abstract/empty, or adding allowed annotation for the method. - line: 89
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setAppointment' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setAppointment' static/final/abstract/empty, or adding allowed annotation for the method. - line: 93
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 97
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 101
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'getBillID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'getBillID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 105
    - Class 'Bill' looks like designed for extension (can be subclassed), but the method 'setBillID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Bill' final or making the method 'setBillID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 109
- BillPreviewController
    - Class 'BillPreviewController' looks like designed for extension (can be subclassed), but the method 'saveSuccessExit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BillPreviewController' final or making the method 'saveSuccessExit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 29
    - Class 'BillPreviewController' looks like designed for extension (can be subclassed), but the method 'fillBillPreview' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'BillPreviewController' final or making the method 'fillBillPreview' static/final/abstract/empty, or adding allowed annotation for the method. - line: 48
- Cashier

- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'getAllNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'getAllNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 150
- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'getWaitingRefunds' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'getWaitingRefunds' static/final/abstract/empty, or adding allowed annotation for the method. - line: 277
- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'makeRefund' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'makeRefund' static/final/abstract/empty, or adding allowed annotation for the method. - line: 293
- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'getNoOfRefunds' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'getNoOfRefunds' static/final/abstract/empty, or adding allowed annotation for the method. - line: 309
- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'getCancelledDocAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'getCancelledDocAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 463
- Class 'Cashier' looks like designed for extension (can be subclassed), but the method 'getCancelledLabAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Cashier' final or making the method 'getCancelledLabAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 483

- CashierController
  - Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'fillLineChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'fillLineChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 87
  - Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'showRefundTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'showRefundTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 209
  - Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'loadNameList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'loadNameList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 260
  - Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'searchPatientBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'searchPatientBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 306
  - Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'clearBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or

making the method 'clearBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 408
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'issueBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'issueBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 430
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'loadRefunds' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'loadRefunds' static/final/abstract/empty, or adding allowed annotation for the method. - line: 505
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'fillPaymentHistory' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'fillPaymentHistory' static/final/abstract/empty, or adding allowed annotation for the method. - line: 512
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'makeHistoryTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'makeHistoryTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 603
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 623
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 693
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'loadProfileData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'loadProfileData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 775
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 955
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1001
- Class 'CashierController' looks like designed for extension (can be subclassed), but the method 'addFocusListener' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CashierController' final or making the method 'addFocusListener' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1119

- CurrentUserSummaryController

- Class 'CurrentUserSummaryController' looks like designed for extension (can be subclassed), but the method 'fillUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CurrentUserSummaryController' final or making the method 'fillUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
- Class 'CurrentUserSummaryController' looks like designed for extension (can be subclassed), but the method 'load' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'CurrentUserSummaryController' final or making the method 'load' static/final/abstract/empty, or adding allowed annotation for the method. - line: 89

- DatabaseOperator
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'connect' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'connect' static/final/abstract/empty, or adding allowed annotation for the method. - line: 112
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'close' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'close' static/final/abstract/empty, or adding allowed annotation for the method. - line: 123
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'createDatabase' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'createDatabase' static/final/abstract/empty, or adding allowed annotation for the method. - line: 128
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showDatabases' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showDatabases' static/final/abstract/empty, or adding allowed annotation for the method. - line: 149
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'useDatabse' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'useDatabse' static/final/abstract/empty, or adding allowed annotation for the method. - line: 174
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'createTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'createTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 197
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showTables' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showTables' static/final/abstract/empty, or adding allowed annotation for the method. - line: 215
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showTableMetaData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showTableMetaData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 239
  - Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'addTableRow' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or

making the method 'addTableRow' static/final/abstract/empty, or adding allowed annotation for the method. - line: 302
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'deleteTableRow' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'deleteTableRow' static/final/abstract/empty, or adding allowed annotation for the method. - line: 346
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'deleteTableRow' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'deleteTableRow' static/final/abstract/empty, or adding allowed annotation for the method. - line: 364
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showTableData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showTableData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 383
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showTableData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showTableData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 438
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'showTableData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'showTableData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 466
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'customSelection' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'customSelection' static/final/abstract/empty, or adding allowed annotation for the method. - line: 500
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'customInsertion' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'customInsertion' static/final/abstract/empty, or adding allowed annotation for the method. - line: 556
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'customDeletion' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'customDeletion' static/final/abstract/empty, or adding allowed annotation for the method. - line: 580
- Class 'DatabaseOperator' looks like designed for extension (can be subclassed), but the method 'deleteTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DatabaseOperator' final or making the method 'deleteTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 595

- Doctor
  - Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getUsername' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getUsername' static/final/abstract/empty, or adding allowed annotation for the method. - line: 43

- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 59
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'updateProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'updateProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 95
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'updateDoctorInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'updateDoctorInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 124
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'updateAccountInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'updateAccountInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 153
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'doctorTimeTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'doctorTimeTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 199
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'removeDoctorTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'removeDoctorTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 221
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'removeDoctorTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'removeDoctorTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 238
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'doctorTimeTableAddSlot' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'doctorTimeTableAddSlot' static/final/abstract/empty, or adding allowed annotation for the method. - line: 255
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 295
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getTestResults' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getTestResults' static/final/abstract/empty, or adding allowed annotation for the method. - line: 316
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getTestResults' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method

'getTestResults' static/final/abstract/empty, or adding allowed annotation for the method. - line: 366

- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getPatientInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getPatientInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 470
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'searchByName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'searchByName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 510
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getAllNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getAllNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 543
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'diagnose' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'diagnose' static/final/abstract/empty, or adding allowed annotation for the method. - line: 561
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'bill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'bill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 599
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getLabFee' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getLabFee' static/final/abstract/empty, or adding allowed annotation for the method. - line: 680
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'allergies' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'allergies' static/final/abstract/empty, or adding allowed annotation for the method. - line: 701
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'prescribe' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'prescribe' static/final/abstract/empty, or adding allowed annotation for the method. - line: 722
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getDrugInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getDrugInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 761
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getDrugGenericInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getDrugGenericInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 784
- Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getDrugBrandInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the

method 'getDrugBrandInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 813
    - Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getPatientAttendence' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getPatientAttendence' static/final/abstract/empty, or adding allowed annotation for the method. - line: 841
    - Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getTodayAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getTodayAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 869
    - Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'nameSuggestor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'nameSuggestor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 888
    - Class 'Doctor' looks like designed for extension (can be subclassed), but the method 'getLabPatientInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Doctor' final or making the method 'getLabPatientInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 956
- DoctorController
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'fillAreaChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'fillAreaChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 153
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'setTodayAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'setTodayAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 212
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'loadDrugList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'loadDrugList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 767
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'loadTestList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'loadTestList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 784
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'loadNameList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'loadNameList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 803
    - Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'checkForBrands' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'checkForBrands' static/final/abstract/empty, or adding allowed annotation for the method. - line: 850

- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'loadProfileData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'loadProfileData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1206
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'MakeAvailabilityTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'MakeAvailabilityTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1297
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1372
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1450
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1499
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1696
- Class 'DoctorController' looks like designed for extension (can be subclassed), but the method 'addFocusListener' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorController' final or making the method 'addFocusListener' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1838

- DoctorDetail
  - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'getDoctorID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'getDoctorID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 30
  - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'setDoctorID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'setDoctorID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
  - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'getDoctorName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'getDoctorName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 38
  - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'setDoctorName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or

making the method 'setDoctorName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'getArea' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'getArea' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'setArea' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'setArea' static/final/abstract/empty, or adding allowed annotation for the method. - line: 50
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'getAvailability' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'getAvailability' static/final/abstract/empty, or adding allowed annotation for the method. - line: 54
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'setAvailability' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'setAvailability' static/final/abstract/empty, or adding allowed annotation for the method. - line: 58
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'getDays' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'getDays' static/final/abstract/empty, or adding allowed annotation for the method. - line: 62
    - Class 'DoctorDetail' looks like designed for extension (can be subclassed), but the method 'setDays' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'DoctorDetail' final or making the method 'setDays' static/final/abstract/empty, or adding allowed annotation for the method. - line: 66
- Drug
    - Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 28
    - Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 32
    - Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getType' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getType' static/final/abstract/empty, or adding allowed annotation for the method. - line: 36
    - Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setType' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setType' static/final/abstract/empty, or adding allowed annotation for the method. - line: 40

- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getUnit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getUnit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 44
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setUnit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setUnit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 48
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getPrice' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getPrice' static/final/abstract/empty, or adding allowed annotation for the method. - line: 52
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setPrice' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setPrice' static/final/abstract/empty, or adding allowed annotation for the method. - line: 56
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getAmount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getAmount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 60
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setAmount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setAmount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 64
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'getSuppliers' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'getSuppliers' static/final/abstract/empty, or adding allowed annotation for the method. - line: 68
- Class 'Drug' looks like designed for extension (can be subclassed), but the method 'setSuppliers' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Drug' final or making the method 'setSuppliers' static/final/abstract/empty, or adding allowed annotation for the method. - line: 72

- ErrorController
  - Class 'ErrorController' looks like designed for extension (can be subclassed), but the method 'addMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ErrorController' final or making the method 'addMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
- LabAssistant
  - Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 31
  - Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'updateProfileInfo' does not have javadoc that explains how to do that safely. If

class is not designed for extension consider making the class 'LabAssistant' final or making the method 'updateProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 67

- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'updateLabAssistantInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'updateLabAssistantInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 95
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'updateAccountInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'updateAccountInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 124
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getLabTestNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getLabTestNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 153
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getLabTestInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getLabTestInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 176
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getPatientDetails' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getPatientDetails' static/final/abstract/empty, or adding allowed annotation for the method. - line: 198
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getPrescriptions' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getPrescriptions' static/final/abstract/empty, or adding allowed annotation for the method. - line: 220
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'lastMonthsAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'lastMonthsAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 242
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'lastMonthsReports' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'lastMonthsReports' static/final/abstract/empty, or adding allowed annotation for the method. - line: 266
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 303
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getTodayAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getTodayAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 318

- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getUrineFullReport' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getUrineFullReport' static/final/abstract/empty, or adding allowed annotation for the method. - line: 791
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getLipidTestReport' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getLipidTestReport' static/final/abstract/empty, or adding allowed annotation for the method. - line: 809
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getBloodGroupingRh' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getBloodGroupingRh' static/final/abstract/empty, or adding allowed annotation for the method. - line: 826
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getCompleteBloodCount' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getCompleteBloodCount' static/final/abstract/empty, or adding allowed annotation for the method. - line: 843
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getLiverFunctionTest' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getLiverFunctionTest' static/final/abstract/empty, or adding allowed annotation for the method. - line: 859
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getRenalFunctionTest' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getRenalFunctionTest' static/final/abstract/empty, or adding allowed annotation for the method. - line: 875
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getSeriumCreatinePhosphokinaseTotal' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getSeriumCreatinePhosphokinaseTotal' static/final/abstract/empty, or adding allowed annotation for the method. - line: 891
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getSeriumCreatinePhosphokinase' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getSeriumCreatinePhosphokinase' static/final/abstract/empty, or adding allowed annotation for the method. - line: 908
- Class 'LabAssistant' looks like designed for extension (can be subclassed), but the method 'getPatientInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistant' final or making the method 'getPatientInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 925

- LabAssistantController
  - Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'fillPieChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'fillPieChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 87
  - Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'setTabsDisabled' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class

'LabAssistantController' final or making the method 'setTabsDisabled' static/final/abstract/empty, or adding allowed annotation for the method. - line: 142

- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'showReport' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'showReport' static/final/abstract/empty, or adding allowed annotation for the method. - line: 154
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'searchAppointment' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'searchAppointment' static/final/abstract/empty, or adding allowed annotation for the method. - line: 228
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'clear' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'clear' static/final/abstract/empty, or adding allowed annotation for the method. - line: 425
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'fillPrescriptionInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'fillPrescriptionInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 439
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'fillLabAppiontments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'fillLabAppiontments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 462
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'fillTodayAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'fillTodayAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 508
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'setAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'setAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 516
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1090
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1139
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'loadProfileData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'loadProfileData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1231
- Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'addFocusListener' does not have javadoc that explains how to do that

safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'addFocusListener' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1424
    - Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1485
    - Class 'LabAssistantController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabAssistantController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1544
- LabReport
    - Class 'LabReport' looks like designed for extension (can be subclassed), but the method 'getConstituent' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabReport' final or making the method 'getConstituent' static/final/abstract/empty, or adding allowed annotation for the method. - line: 21
    - Class 'LabReport' looks like designed for extension (can be subclassed), but the method 'setConstituent' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabReport' final or making the method 'setConstituent' static/final/abstract/empty, or adding allowed annotation for the method. - line: 25
    - Class 'LabReport' looks like designed for extension (can be subclassed), but the method 'getResult' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabReport' final or making the method 'getResult' static/final/abstract/empty, or adding allowed annotation for the method. - line: 29
    - Class 'LabReport' looks like designed for extension (can be subclassed), but the method 'setResult' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabReport' final or making the method 'setResult' static/final/abstract/empty, or adding allowed annotation for the method. - line: 33
- LabReportPreviewController
    - Class 'LabReportPreviewController' looks like designed for extension (can be subclassed), but the method 'setData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LabReportPreviewController' final or making the method 'setData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42
- LoginController
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadDoctor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or making the method 'loadDoctor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 104
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadPharmacist' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or making the method 'loadPharmacist' static/final/abstract/empty, or adding allowed annotation for the method. - line: 131
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadReceptionist' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or

making the method 'loadReceptionist' static/final/abstract/empty, or adding allowed annotation for the method. - line: 156
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadCashier' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or making the method 'loadCashier' static/final/abstract/empty, or adding allowed annotation for the method. - line: 179
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadAdmin' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or making the method 'loadAdmin' static/final/abstract/empty, or adding allowed annotation for the method. - line: 204
    - Class 'LoginController' looks like designed for extension (can be subclassed), but the method 'loadLabAssistant' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'LoginController' final or making the method 'loadLabAssistant' static/final/abstract/empty, or adding allowed annotation for the method. - line: 232
- MainApp
    - Class 'MainApp' looks like designed for extension (can be subclassed), but the method 'start' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'MainApp' final or making the method 'start' static/final/abstract/empty, or adding allowed annotation for the method. - line: 23
- Message
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'getSender' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'getSender' static/final/abstract/empty, or adding allowed annotation for the method. - line: 22
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'setSender' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'setSender' static/final/abstract/empty, or adding allowed annotation for the method. - line: 26
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'getSubject' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'getSubject' static/final/abstract/empty, or adding allowed annotation for the method. - line: 30
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'setSubject' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'setSubject' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'getMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'getMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 38
    - Class 'Message' looks like designed for extension (can be subclassed), but the method 'setMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Message' final or making the method 'setMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42
- NewDoctorTimeSlotController

- Class 'NewDoctorTimeSlotController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'NewDoctorTimeSlotController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 67

- NewMessageController
  - Class 'NewMessageController' looks like designed for extension (can be subclassed), but the method 'loadUserNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'NewMessageController' final or making the method 'loadUserNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 59
  - Class 'NewMessageController' looks like designed for extension (can be subclassed), but the method 'load' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'NewMessageController' final or making the method 'load' static/final/abstract/empty, or adding allowed annotation for the method. - line: 156
  - Class 'NewMessageController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'NewMessageController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 184
- PatientAccountSuccessController
  - Class 'PatientAccountSuccessController' looks like designed for extension (can be subclassed), but the method 'fillPatientData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientAccountSuccessController' final or making the method 'fillPatientData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 31
  - Class 'PatientAccountSuccessController' looks like designed for extension (can be subclassed), but the method 'saveSuccessExit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PatientAccountSuccessController' final or making the method 'saveSuccessExit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 44
- Pharmacist
  - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 50
  - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getPrescriptionInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getPrescriptionInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 85
  - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getPrescribedDoc' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getPrescribedDoc' static/final/abstract/empty, or adding allowed annotation for the method. - line: 99
  - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getpharmacyHistory' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getpharmacyHistory' static/final/abstract/empty, or adding allowed annotation for the method. - line: 128

- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 151
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getAllNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getAllNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 184
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'addNewDrug' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'addNewDrug' static/final/abstract/empty, or adding allowed annotation for the method. - line: 202
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'addNewStock' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'addNewStock' static/final/abstract/empty, or adding allowed annotation for the method. - line: 261
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getStockInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getStockInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 320
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getStockSummary' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getStockSummary' static/final/abstract/empty, or adding allowed annotation for the method. - line: 341
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getStockSummary2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getStockSummary2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 394
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getSupplierNames2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getSupplierNames2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 522
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugPrices' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugPrices' static/final/abstract/empty, or adding allowed annotation for the method. - line: 538
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getGenericNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getGenericNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 564
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getBrandNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the

method 'getBrandNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 580

- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugGenericInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugGenericInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 597
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 624
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugAmounts' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugAmounts' static/final/abstract/empty, or adding allowed annotation for the method. - line: 638
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'getDrugStockID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'getDrugStockID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 653
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'reduceStock' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'reduceStock' static/final/abstract/empty, or adding allowed annotation for the method. - line: 670
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'checkForGenName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'checkForGenName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 687
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'addNewDrug2' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'addNewDrug2' static/final/abstract/empty, or adding allowed annotation for the method. - line: 703
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'checkForBrandName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'checkForBrandName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 741
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'addNewBrand' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'addNewBrand' static/final/abstract/empty, or adding allowed annotation for the method. - line: 757
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'checkForSupplierName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'checkForSupplierName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 795
- Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'addNewSupplier' does not have javadoc that explains how to do that safely. If class is

not designed for extension consider making the class 'Pharmacist' final or making the method 'addNewSupplier' static/final/abstract/empty, or adding allowed annotation for the method. - line: 811
    - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'updateStock' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'updateStock' static/final/abstract/empty, or adding allowed annotation for the method. - line: 850
    - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'updateProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'updateProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 890
    - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'updatePharmacistInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'updatePharmacistInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 918
    - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'updateAccountInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'updateAccountInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 947
    - Class 'Pharmacist' looks like designed for extension (can be subclassed), but the method 'bill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Pharmacist' final or making the method 'bill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 975
- PharmacistController
    - Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'loadNameList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'loadNameList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 361
    - Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'makeStockTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'makeStockTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 557
    - Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'fillPieChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'fillPieChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 567
    - Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'fillBarChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'fillBarChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 605
    - Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 726

- Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 796
- Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'loadProfileData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'loadProfileData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 885
- Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1120
- Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1179
- Class 'PharmacistController' looks like designed for extension (can be subclassed), but the method 'addFocusListener' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PharmacistController' final or making the method 'addFocusListener' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1297

- Popover2Controller
  - Class 'Popover2Controller' looks like designed for extension (can be subclassed), but the method 'fillDaysList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Popover2Controller' final or making the method 'fillDaysList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 31
  - Class 'Popover2Controller' looks like designed for extension (can be subclassed), but the method 'closePopUp' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Popover2Controller' final or making the method 'closePopUp' static/final/abstract/empty, or adding allowed annotation for the method. - line: 36
- PopoverController
  - Class 'PopoverController' looks like designed for extension (can be subclassed), but the method 'fillBrandList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PopoverController' final or making the method 'fillBrandList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
  - Class 'PopoverController' looks like designed for extension (can be subclassed), but the method 'close' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PopoverController' final or making the method 'close' static/final/abstract/empty, or adding allowed annotation for the method. - line: 73
- PopupAskController
  - Class 'PopupAskController' looks like designed for extension (can be subclassed), but the method 'message' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PopupAskController' final or making the method 'message' static/final/abstract/empty, or adding allowed annotation for the method. - line: 92
- Prescription

- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'getPrescID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'getPrescID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 25
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'setPrescID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'setPrescID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 29
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'getDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'getDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 33
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'setDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'setDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 37
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'getDoctor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'getDoctor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 41
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'setDoctor' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'setDoctor' static/final/abstract/empty, or adding allowed annotation for the method. - line: 45
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'getPrescription' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'getPrescription' static/final/abstract/empty, or adding allowed annotation for the method. - line: 49
- Class 'Prescription' looks like designed for extension (can be subclassed), but the method 'setPrescription' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Prescription' final or making the method 'setPrescription' static/final/abstract/empty, or adding allowed annotation for the method. - line: 53

- PrescriptionListController
  - Class 'PrescriptionListController' looks like designed for extension (can be subclassed), but the method 'fillTableData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'PrescriptionListController' final or making the method 'fillTableData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 36
- ReadMessageController
  - Class 'ReadMessageController' looks like designed for extension (can be subclassed), but the method 'fillMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReadMessageController' final or making the method 'fillMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 131
  - Class 'ReadMessageController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do

that safely. If class is not designed for extension consider making the class 'ReadMessageController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 150

- Receptionist
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'updateProfileInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'updateProfileInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 72
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'updateAccountInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'updateAccountInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 104
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getDoctorTimeTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getDoctorTimeTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 132
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getPatientInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getPatientInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 154
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'updatePatientInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'updatePatientInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 262
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getLabTestInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getLabTestInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 281
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 302
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'doctorAppointmentAvailableTime' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'doctorAppointmentAvailableTime' static/final/abstract/empty, or adding allowed annotation for the method. - line: 675
  - Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getAppointmentDetails' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getAppointmentDetails' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1156

- Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'refund' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'refund' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1201
- Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getDoctorDetails' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getDoctorDetails' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1350
- Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getDocAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getDocAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1406
- Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getLabAppointments' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getLabAppointments' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1426
- Class 'Receptionist' looks like designed for extension (can be subclassed), but the method 'getAllNames' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Receptionist' final or making the method 'getAllNames' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1446

- ReceptionistController
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'fillLineChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'fillLineChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 81
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'makeSummaryTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'makeSummaryTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 249
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'fillCurrentDoctors' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'fillCurrentDoctors' static/final/abstract/empty, or adding allowed annotation for the method. - line: 259
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'patientAdd' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'patientAdd' static/final/abstract/empty, or adding allowed annotation for the method. - line: 485
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'loadNameList' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'loadNameList' static/final/abstract/empty, or adding allowed annotation for the method. - line: 627
  - Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'fillConsultationAreas' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class

'ReceptionistController' final or making the method 'fillConsultationAreas' static/final/abstract/empty, or adding allowed annotation for the method. - line: 713
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'selectDoctors' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'selectDoctors' static/final/abstract/empty, or adding allowed annotation for the method. - line: 737
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'setDates' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'setDates' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1112
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'loadProfileData' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'loadProfileData' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1161
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'editProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'editProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1394
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'loadProfileImage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'loadProfileImage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1443
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'showAppointmentSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'showAppointmentSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1456
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'showPatientAccountSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'showPatientAccountSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1480
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1504
- Class 'ReceptionistController' looks like designed for extension (can be subclassed), but the method 'setPaceholders' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReceptionistController' final or making the method 'setPaceholders' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1549

- Refund
  - Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getPatientID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method

'getPatientID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 31

- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setPatientID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'setPatientID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 35
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'getDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 39
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setDate' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'setDate' static/final/abstract/empty, or adding allowed annotation for the method. - line: 43
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getBillID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'getBillID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 47
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setBillID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'setBillID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 51
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getService' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'getService' static/final/abstract/empty, or adding allowed annotation for the method. - line: 55
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setService' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'setService' static/final/abstract/empty, or adding allowed annotation for the method. - line: 59
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'getBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 63
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setBill' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'setBill' static/final/abstract/empty, or adding allowed annotation for the method. - line: 67
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'getRefund' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'Refund' final or making the method 'getRefund' static/final/abstract/empty, or adding allowed annotation for the method. - line: 71
- Class 'Refund' looks like designed for extension (can be subclassed), but the method 'setRefund' does not have javadoc that explains how to do that safely. If class is not

designed for extension consider making the class 'Refund' final or making the method 'setRefund' static/final/abstract/empty, or adding allowed annotation for the method. - line: 75

- RefundController
  - Class 'RefundController' looks like designed for extension (can be subclassed), but the method 'fillRefundTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'RefundController' final or making the method 'fillRefundTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
  - Class 'RefundController' looks like designed for extension (can be subclassed), but the method 'closeRefundTable' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'RefundController' final or making the method 'closeRefundTable' static/final/abstract/empty, or adding allowed annotation for the method. - line: 116
- ReportsController
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillPatientAttendence' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillPatientAttendence' static/final/abstract/empty, or adding allowed annotation for the method. - line: 64
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillPieChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillPieChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 155
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillAppointmentChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillAppointmentChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 206
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillCancelledAppointmentChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillCancelledAppointmentChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 360
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillStockChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillStockChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 514
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillSupplierChart' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillSupplierChart' static/final/abstract/empty, or adding allowed annotation for the method. - line: 625
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillTotalIncomeBarGraph' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'fillTotalIncomeBarGraph' static/final/abstract/empty, or adding allowed annotation for the method. - line: 679
  - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'fillIcome' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or

making the method 'fillIcome' static/final/abstract/empty, or adding allowed annotation for the method. - line: 786
    - Class 'ReportsController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'ReportsController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 1053
- SettingsController
    - Class 'SettingsController' looks like designed for extension (can be subclassed), but the method 'loadConfigFile' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SettingsController' final or making the method 'loadConfigFile' static/final/abstract/empty, or adding allowed annotation for the method. - line: 56
    - Class 'SettingsController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SettingsController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 299
- SuccessIndicatorController
    - Class 'SuccessIndicatorController' looks like designed for extension (can be subclassed), but the method 'saveSuccessExit' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SuccessIndicatorController' final or making the method 'saveSuccessExit' static/final/abstract/empty, or adding allowed annotation for the method. - line: 29
- SysUserController
    - Class 'SysUserController' looks like designed for extension (can be subclassed), but the method 'loadInfo' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SysUserController' final or making the method 'loadInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 64
    - Class 'SysUserController' looks like designed for extension (can be subclassed), but the method 'load' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SysUserController' final or making the method 'load' static/final/abstract/empty, or adding allowed annotation for the method. - line: 182
    - Class 'SysUserController' looks like designed for extension (can be subclassed), but the method 'showSuccessIndicator' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'SysUserController' final or making the method 'showSuccessIndicator' static/final/abstract/empty, or adding allowed annotation for the method. - line: 188
- User
    - Class 'User' looks like designed for extension (can be subclassed), but the method 'checkUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'checkUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 126
    - Class 'User' looks like designed for extension (can be subclassed), but the method 'saveLogin' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'saveLogin' static/final/abstract/empty, or adding allowed annotation for the method. - line: 142
    - Class 'User' looks like designed for extension (can be subclassed), but the method 'saveLogout' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method

'saveLogout' static/final/abstract/empty, or adding allowed annotation for the method. - line: 158

- Class 'User' looks like designed for extension (can be subclassed), but the method 'sendMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'sendMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 174
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getMessages' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getMessages' static/final/abstract/empty, or adding allowed annotation for the method. - line: 220
- Class 'User' looks like designed for extension (can be subclassed), but the method 'deleteMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'deleteMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 238
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 253
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 274
- Class 'User' looks like designed for extension (can be subclassed), but the method 'setProfilePic' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'setProfilePic' static/final/abstract/empty, or adding allowed annotation for the method. - line: 296
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getName' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getName' static/final/abstract/empty, or adding allowed annotation for the method. - line: 314
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getUserNameAndID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getUserNameAndID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 332
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getCurrentUserNameAndID' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'getCurrentUserNameAndID' static/final/abstract/empty, or adding allowed annotation for the method. - line: 352
- Class 'User' looks like designed for extension (can be subclassed), but the method 'setMessageRead' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'User' final or making the method 'setMessageRead' static/final/abstract/empty, or adding allowed annotation for the method. - line: 370
- Class 'User' looks like designed for extension (can be subclassed), but the method 'getMessageSenderInfo' does not have javadoc that explains how to do that safely. If

class is not designed for extension consider making the class 'User' final or making the method 'getMessageSenderInfo' static/final/abstract/empty, or adding allowed annotation for the method. - line: 387

- UserAccount
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'setUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'setUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 30
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'setLastLogin' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'setLastLogin' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'setOnline' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'setOnline' static/final/abstract/empty, or adding allowed annotation for the method. - line: 38
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'getUser' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'getUser' static/final/abstract/empty, or adding allowed annotation for the method. - line: 42
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'getLastLogin' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'getLastLogin' static/final/abstract/empty, or adding allowed annotation for the method. - line: 46
  - Class 'UserAccount' looks like designed for extension (can be subclassed), but the method 'getOnline' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccount' final or making the method 'getOnline' static/final/abstract/empty, or adding allowed annotation for the method. - line: 50
- UserAccountController
  - Class 'UserAccountController' looks like designed for extension (can be subclassed), but the method 'fillUserDetail' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'UserAccountController' final or making the method 'fillUserDetail' static/final/abstract/empty, or adding allowed annotation for the method. - line: 357
- WarningController
  - Class 'WarningController' looks like designed for extension (can be subclassed), but the method 'addMessage' does not have javadoc that explains how to do that safely. If class is not designed for extension consider making the class 'WarningController' final or making the method 'addMessage' static/final/abstract/empty, or adding allowed annotation for the method. - line: 34

- Magic Number
  - Admin
    - '3' is a magic number. - line: 169
    - '3' is a magic number. - line: 170
    - '5' is a magic number. - line: 177
    - '3' is a magic number. - line: 194
    - '3' is a magic number. - line: 195
    - '4' is a magic number. - line: 202

- '4' is a magic number. - line: 220
- '4' is a magic number. - line: 221
- '3' is a magic number. - line: 228
- '3' is a magic number. - line: 272
- '3' is a magic number. - line: 273
- '4' is a magic number. - line: 280
- '3' is a magic number. - line: 301
- '3' is a magic number. - line: 302
- '4' is a magic number. - line: 309
- '8' is a magic number. - line: 671

- AdminController
  - '6' is a magic number. - line: 132
  - '12' is a magic number. - line: 640
  - '12' is a magic number. - line: 649
  - '500' is a magic number. - line: 733
  - '500' is a magic number. - line: 749
  - '3' is a magic number. - line: 869
  - '3' is a magic number. - line: 1014
  - '3' is a magic number. - line: 1015
  - '3' is a magic number. - line: 1026
  - '4' is a magic number. - line: 1099
  - '3' is a magic number. - line: 1181
  - '100' is a magic number. - line: 1197
  - '320' is a magic number. - line: 1197
  - '320' is a magic number. - line: 1198
  - '100' is a magic number. - line: 1198
  - '4' is a magic number. - line: 1438
  - '7' is a magic number. - line: 1439
  - '5' is a magic number. - line: 1439
  - '10' is a magic number. - line: 1440
  - '8' is a magic number. - line: 1440
  - '500' is a magic number. - line: 1780
  - '500' is a magic number. - line: 1818
  - '500' is a magic number. - line: 1835
  - '500' is a magic number. - line: 1851
- AllAppointmentsController
  - '4' is a magic number. - line: 78
  - '4' is a magic number. - line: 79
  - '3' is a magic number. - line: 80
  - '3' is a magic number. - line: 81
- AllMessagesController
  - '3' is a magic number. - line: 61
  - '4' is a magic number. - line: 62
  - '5' is a magic number. - line: 63
  - '25' is a magic number. - line: 74
  - '25' is a magic number. - line: 75
  - '35' is a magic number. - line: 92
  - '35' is a magic number. - line: 93
  - '8' is a magic number. - line: 106
  - '1.5' is a magic number. - line: 111
  - '8' is a magic number. - line: 116
- Cashier
  - '3' is a magic number. - line: 97
  - '3' is a magic number. - line: 98

- '4' is a magic number. - line: 104
- '3' is a magic number. - line: 124
- '3' is a magic number. - line: 126
- '4' is a magic number. - line: 232
- '3' is a magic number. - line: 252
- '3' is a magic number. - line: 254

- CashierController
  - '5' is a magic number. - line: 193
  - '5' is a magic number. - line: 194
  - '5' is a magic number. - line: 194
  - '5' is a magic number. - line: 196
  - '5' is a magic number. - line: 197
  - '5' is a magic number. - line: 197
  - '3' is a magic number. - line: 276
  - '4' is a magic number. - line: 296
  - '3' is a magic number. - line: 298
  - '5' is a magic number. - line: 346
  - '3' is a magic number. - line: 347
  - '4' is a magic number. - line: 348
  - '10' is a magic number. - line: 370
  - '11' is a magic number. - line: 370
  - '8' is a magic number. - line: 374
  - '5' is a magic number. - line: 382
  - '100' is a magic number. - line: 382
  - '100' is a magic number. - line: 514
  - '100' is a magic number. - line: 534
  - '5' is a magic number. - line: 534
  - '150' is a magic number. - line: 565
  - '10' is a magic number. - line: 573
  - '3' is a magic number. - line: 576
  - '5' is a magic number. - line: 577
  - '4' is a magic number. - line: 577
  - '6' is a magic number. - line: 578
  - '7' is a magic number. - line: 578
  - '8' is a magic number. - line: 579
  - '7' is a magic number. - line: 586
  - '7' is a magic number. - line: 587
  - '7' is a magic number. - line: 599
  - '150' is a magic number. - line: 606
  - '4' is a magic number. - line: 799
  - '5' is a magic number. - line: 800
  - '7' is a magic number. - line: 800
  - '10' is a magic number. - line: 801
  - '8' is a magic number. - line: 801
  - '9' is a magic number. - line: 1037
  - '500' is a magic number. - line: 1042
  - '9' is a magic number. - line: 1048
  - '500' is a magic number. - line: 1062
  - '500' is a magic number. - line: 1079
  - '500' is a magic number. - line: 1095
  - '500' is a magic number. - line: 1111
- CurrentUserSummaryController
  - '3' is a magic number. - line: 60
- DatabaseOperator

- '4' is a magic number. - line: 256
- '4' is a magic number. - line: 263

- Doctor
  - '4' is a magic number. - line: 276
  - '4' is a magic number. - line: 409
  - '3' is a magic number. - line: 422
  - '3' is a magic number. - line: 571
  - '3' is a magic number. - line: 572
  - '4' is a magic number. - line: 578
  - '3' is a magic number. - line: 626
  - '3' is a magic number. - line: 627
  - '4' is a magic number. - line: 633
  - '3' is a magic number. - line: 656
  - '3' is a magic number. - line: 658
  - '4' is a magic number. - line: 732
  - '4' is a magic number. - line: 733
  - '5' is a magic number. - line: 739
- DoctorController
  - '5' is a magic number. - line: 109
  - '30' is a magic number. - line: 116
  - '30' is a magic number. - line: 118
  - '60' is a magic number. - line: 122
  - '30' is a magic number. - line: 122
  - '7' is a magic number. - line: 299
  - '8' is a magic number. - line: 300
  - '4' is a magic number. - line: 304
  - '3' is a magic number. - line: 312
  - '9' is a magic number. - line: 315
  - '3' is a magic number. - line: 421
  - '3' is a magic number. - line: 422
  - '3' is a magic number. - line: 470
  - '10' is a magic number. - line: 540
  - '10' is a magic number. - line: 541
  - '10' is a magic number. - line: 563
  - '3' is a magic number. - line: 705
  - '3' is a magic number. - line: 743
  - '1.8' is a magic number. - line: 748
  - '25.0' is a magic number. - line: 751
  - '3' is a magic number. - line: 819
  - '4' is a magic number. - line: 837
  - '3' is a magic number. - line: 839
  - '30' is a magic number. - line: 875
  - '4' is a magic number. - line: 1230
  - '7' is a magic number. - line: 1231
  - '5' is a magic number. - line: 1231
  - '10' is a magic number. - line: 1232
  - '8' is a magic number. - line: 1232
  - '8' is a magic number. - line: 1280
  - '8' is a magic number. - line: 1281
  - '8' is a magic number. - line: 1293
  - '5' is a magic number. - line: 1546
  - '1000' is a magic number. - line: 1546
  - '500' is a magic number. - line: 1746
  - '9' is a magic number. - line: 1758

- '500' is a magic number. - line: 1763
- '500' is a magic number. - line: 1781
- '500' is a magic number. - line: 1798
- '500' is a magic number. - line: 1814
- '500' is a magic number. - line: 1830

- LabAssistant
  - '6' is a magic number. - line: 250
  - '30' is a magic number. - line: 252
  - '30' is a magic number. - line: 284
  - '8' is a magic number. - line: 288
  - '4' is a magic number. - line: 372
  - '4' is a magic number. - line: 426
  - '4' is a magic number. - line: 480
  - '3' is a magic number. - line: 544
  - '4' is a magic number. - line: 550
  - '4' is a magic number. - line: 606
  - '4' is a magic number. - line: 650
  - '4' is a magic number. - line: 656
  - '3' is a magic number. - line: 700
  - '4' is a magic number. - line: 706
  - '4' is a magic number. - line: 766
- LabAssistantController
  - '12' is a magic number. - line: 90
  - '3' is a magic number. - line: 175
  - '4' is a magic number. - line: 190
  - '3' is a magic number. - line: 195
  - '4' is a magic number. - line: 240
  - '3' is a magic number. - line: 264
  - '4' is a magic number. - line: 269
  - '5' is a magic number. - line: 274
  - '6' is a magic number. - line: 279
  - '7' is a magic number. - line: 284
  - '3' is a magic number. - line: 296
  - '3' is a magic number. - line: 302
  - '4' is a magic number. - line: 304
  - '7' is a magic number. - line: 305
  - '5' is a magic number. - line: 305
  - '10' is a magic number. - line: 306
  - '8' is a magic number. - line: 306
  - '3' is a magic number. - line: 343
  - '4' is a magic number. - line: 358
  - '3' is a magic number. - line: 363
  - '3' is a magic number. - line: 468
  - '4' is a magic number. - line: 469
  - '5' is a magic number. - line: 470
  - '6' is a magic number. - line: 471
  - '7' is a magic number. - line: 472
  - '8' is a magic number. - line: 473
  - '9' is a magic number. - line: 474
  - '10' is a magic number. - line: 475
  - '11' is a magic number. - line: 476
  - '12' is a magic number. - line: 477
  - '12' is a magic number. - line: 495
  - '5' is a magic number. - line: 533

        - '30' is a magic number. - line: 540
        - '30' is a magic number. - line: 542
        - '30' is a magic number. - line: 546
        - '60' is a magic number. - line: 546
        - '4' is a magic number. - line: 1255
        - '7' is a magic number. - line: 1256
        - '5' is a magic number. - line: 1256
        - '8' is a magic number. - line: 1257
        - '10' is a magic number. - line: 1257
    - LabReportPreviewController
        - '3' is a magic number. - line: 137
    - NewMessageController
        - '3' is a magic number. - line: 70
        - '3' is a magic number. - line: 71
        - '3' is a magic number. - line: 86
        - '3' is a magic number. - line: 169
    - Pharmacist
        - '4' is a magic number. - line: 220
        - '3' is a magic number. - line: 239
        - '3' is a magic number. - line: 241
        - '3' is a magic number. - line: 272
        - '3' is a magic number. - line: 273
        - '4' is a magic number. - line: 279
        - '5' is a magic number. - line: 296
        - '6' is a magic number. - line: 296
        - '5' is a magic number. - line: 298
        - '6' is a magic number. - line: 298
        - '4' is a magic number. - line: 721
        - '4' is a magic number. - line: 775
        - '3' is a magic number. - line: 822
        - '3' is a magic number. - line: 823
        - '4' is a magic number. - line: 829
        - '3' is a magic number. - line: 861
        - '3' is a magic number. - line: 862
        - '4' is a magic number. - line: 868
        - '3' is a magic number. - line: 1004
        - '3' is a magic number. - line: 1005
        - '4' is a magic number. - line: 1011
        - '3' is a magic number. - line: 1034
        - '3' is a magic number. - line: 1036
    - PharmacistController
        - '3' is a magic number. - line: 377
        - '4' is a magic number. - line: 394
        - '3' is a magic number. - line: 396
        - '3' is a magic number. - line: 531
        - '4' is a magic number. - line: 532
        - '5' is a magic number. - line: 533
        - '6' is a magic number. - line: 534
        - '7' is a magic number. - line: 540
        - '7' is a magic number. - line: 541
        - '7' is a magic number. - line: 553
        - '3' is a magic number. - line: 620
        - '5' is a magic number. - line: 640
        - '5' is a magic number. - line: 640

        - '4' is a magic number. - line: 909
        - '7' is a magic number. - line: 910
        - '5' is a magic number. - line: 910
        - '8' is a magic number. - line: 911
        - '10' is a magic number. - line: 911
        - '9' is a magic number. - line: 1217
        - '500' is a magic number. - line: 1222
        - '500' is a magic number. - line: 1240
        - '500' is a magic number. - line: 1257
        - '500' is a magic number. - line: 1273
        - '500' is a magic number. - line: 1289
    - PrescriptionListController
        - '4' is a magic number. - line: 45
        - '3' is a magic number. - line: 45
    - Receptionist
        - '3' is a magic number. - line: 191
        - '3' is a magic number. - line: 192
        - '4' is a magic number. - line: 198
        - '3' is a magic number. - line: 209
        - '3' is a magic number. - line: 210
        - '5' is a magic number. - line: 216
        - '7' is a magic number. - line: 338
        - '7' is a magic number. - line: 341
        - '3' is a magic number. - line: 352
        - '3' is a magic number. - line: 353
        - '3' is a magic number. - line: 359
        - '6' is a magic number. - line: 367
        - '7' is a magic number. - line: 367
        - '3' is a magic number. - line: 367
        - '5' is a magic number. - line: 367
        - '4' is a magic number. - line: 367
        - '7' is a magic number. - line: 382
        - '5' is a magic number. - line: 397
        - '3' is a magic number. - line: 444
        - '4' is a magic number. - line: 450
        - '3' is a magic number. - line: 473
        - '3' is a magic number. - line: 475
        - '4' is a magic number. - line: 526
        - '4' is a magic number. - line: 527
        - '3' is a magic number. - line: 533
        - '5' is a magic number. - line: 547
        - '6' is a magic number. - line: 547
        - '3' is a magic number. - line: 547
        - '4' is a magic number. - line: 547
        - '7' is a magic number. - line: 547
        - '7' is a magic number. - line: 562
        - '5' is a magic number. - line: 577
        - '3' is a magic number. - line: 617
        - '4' is a magic number. - line: 623
        - '3' is a magic number. - line: 646
        - '3' is a magic number. - line: 648
        - '7' is a magic number. - line: 714
        - '60000' is a magic number. - line: 731
        - '5' is a magic number. - line: 734

- '5' is a magic number. - line: 740
- '4' is a magic number. - line: 1218
- '3' is a magic number. - line: 1238
- '3' is a magic number. - line: 1240

- ReceptionistController
  - '5' is a magic number. - line: 190
  - '5' is a magic number. - line: 190
  - '5' is a magic number. - line: 193
  - '5' is a magic number. - line: 193
  - '3' is a magic number. - line: 221
  - '4' is a magic number. - line: 224
  - '4' is a magic number. - line: 225
  - '4' is a magic number. - line: 227
  - '5' is a magic number. - line: 228
  - '7' is a magic number. - line: 232
  - '7' is a magic number. - line: 233
  - '7' is a magic number. - line: 245
  - '7' is a magic number. - line: 304
  - '8' is a magic number. - line: 305
  - '3' is a magic number. - line: 307
  - '6' is a magic number. - line: 315
  - '9' is a magic number. - line: 316
  - '5' is a magic number. - line: 318
  - '4' is a magic number. - line: 326
  - '4' is a magic number. - line: 328
  - '5' is a magic number. - line: 329
  - '7' is a magic number. - line: 329
  - '10' is a magic number. - line: 330
  - '8' is a magic number. - line: 330
  - '7' is a magic number. - line: 350
  - '8' is a magic number. - line: 351
  - '3' is a magic number. - line: 353
  - '6' is a magic number. - line: 361
  - '9' is a magic number. - line: 362
  - '5' is a magic number. - line: 364
  - '4' is a magic number. - line: 372
  - '4' is a magic number. - line: 374
  - '5' is a magic number. - line: 375
  - '7' is a magic number. - line: 375
  - '8' is a magic number. - line: 376
  - '10' is a magic number. - line: 376
  - '3' is a magic number. - line: 643
  - '3' is a magic number. - line: 796
  - '4' is a magic number. - line: 797
  - '5' is a magic number. - line: 798
  - '6' is a magic number. - line: 799
  - '7' is a magic number. - line: 813
  - '7' is a magic number. - line: 827
  - '7' is a magic number. - line: 918
  - '7' is a magic number. - line: 920
  - '3' is a magic number. - line: 972
  - '3' is a magic number. - line: 977
  - '3' is a magic number. - line: 987
  - '3' is a magic number. - line: 1031

- '13' is a magic number. - line: 1098
- '4' is a magic number. - line: 1185
- '5' is a magic number. - line: 1186
- '7' is a magic number. - line: 1186
- '8' is a magic number. - line: 1187
- '10' is a magic number. - line: 1187
- '9' is a magic number. - line: 1587
- '500' is a magic number. - line: 1592
- '9' is a magic number. - line: 1598
- '9' is a magic number. - line: 1610
- '500' is a magic number. - line: 1616
- '9' is a magic number. - line: 1626
- '500' is a magic number. - line: 1642
- '500' is a magic number. - line: 1662
- '500' is a magic number. - line: 1680
- '500' is a magic number. - line: 1700
- '4' is a magic number. - line: 1711
- '500' is a magic number. - line: 1736
- '500' is a magic number. - line: 1753
- '500' is a magic number. - line: 1770

- RefundController
  - '3' is a magic number. - line: 57
  - '4' is a magic number. - line: 58
  - '5' is a magic number. - line: 59
  - '25' is a magic number. - line: 63
  - '25' is a magic number. - line: 64
  - '5' is a magic number. - line: 82
- ReportsController
  - '3' is a magic number. - line: 124
  - '5' is a magic number. - line: 129
  - '5' is a magic number. - line: 129
  - '13' is a magic number. - line: 182
  - '5' is a magic number. - line: 307
  - '5' is a magic number. - line: 307
  - '5' is a magic number. - line: 310
  - '5' is a magic number. - line: 310
  - '5' is a magic number. - line: 464
  - '5' is a magic number. - line: 464
  - '5' is a magic number. - line: 467
  - '5' is a magic number. - line: 467
  - '3' is a magic number. - line: 525
  - '5' is a magic number. - line: 545
  - '5' is a magic number. - line: 545
  - '12' is a magic number. - line: 696
  - '12' is a magic number. - line: 751
  - '12' is a magic number. - line: 803
  - '12' is a magic number. - line: 976
- SysUserController
  - '13' is a magic number. - line: 68
  - '7' is a magic number. - line: 92
  - '8' is a magic number. - line: 93
  - '5' is a magic number. - line: 94
  - '4' is a magic number. - line: 97
  - '6' is a magic number. - line: 103

- '9' is a magic number. - line: 104
- '13' is a magic number. - line: 106
- '12' is a magic number. - line: 120
- '15' is a magic number. - line: 123

- User
  - '3' is a magic number. - line: 185
  - '3' is a magic number. - line: 186
  - '5' is a magic number. - line: 192
- UserAccountController
  - '4' is a magic number. - line: 118
  - '4' is a magic number. - line: 119
  - '3' is a magic number. - line: 125
  - '4' is a magic number. - line: 126
  - '7' is a magic number. - line: 130
  - '8' is a magic number. - line: 137
  - '3' is a magic number. - line: 154
  - '4' is a magic number. - line: 155
  - '7' is a magic number. - line: 159
  - '8' is a magic number. - line: 166
  - '3' is a magic number. - line: 181
  - '4' is a magic number. - line: 182
  - '7' is a magic number. - line: 186
  - '8' is a magic number. - line: 193
  - '3' is a magic number. - line: 206
  - '3' is a magic number. - line: 206
  - '3' is a magic number. - line: 207
  - '3' is a magic number. - line: 208
  - '3' is a magic number. - line: 208
  - '3' is a magic number. - line: 209
  - '4' is a magic number. - line: 209
  - '3' is a magic number. - line: 213
  - '7' is a magic number. - line: 213
  - '3' is a magic number. - line: 220
  - '8' is a magic number. - line: 220
  - '4' is a magic number. - line: 341
  - '4' is a magic number. - line: 345
  - '4' is a magic number. - line: 348
- Validate
  - '10' is a magic number. - line: 19
  - '9' is a magic number. - line: 21
  - '5' is a magic number. - line: 28
  - '5' is a magic number. - line: 29
  - '9' is a magic number. - line: 29
  - '500' is a magic number. - line: 33
  - '500' is a magic number. - line: 36
  - '10' is a magic number. - line: 57
  - '57' is a magic number. - line: 64
  - '48' is a magic number. - line: 64
  - '9' is a magic number. - line: 81
  - '3' is a magic number. - line: 83
  - '7' is a magic number. - line: 85
  - '9' is a magic number. - line: 85
  - '9' is a magic number. - line: 116
  - '3' is a magic number. - line: 118

        - '9' is a magic number. - line: 120
        - '7' is a magic number. - line: 120
        - '6' is a magic number. - line: 155
        - '3' is a magic number. - line: 157
        - '3' is a magic number. - line: 159
        - '6' is a magic number. - line: 159
        - '7' is a magic number. - line: 167
        - '4' is a magic number. - line: 169
        - '7' is a magic number. - line: 171
        - '4' is a magic number. - line: 171
        - '7' is a magic number. - line: 178
    - String Literal Equality
      - Admin
        - Literal Strings should be compared using equals(), not '==='. - line: 342
        - Literal Strings should be compared using equals(), not '==='. - line: 342
        - Literal Strings should be compared using equals(), not '==='. - line: 342
      - Doctor
        - Literal Strings should be compared using equals(), not '==='. - line: 749
        - Literal Strings should be compared using equals(), not '==='. - line: 750
- Usability
  - Hidden Field
    - AddNewDrugController
      - 'pharmacistC' hides a field. - line: 29
      - 'pharmacist' hides a field. - line: 29
      - 'manuDate' hides a field. - line: 245
      - 'expDate' hides a field. - line: 246
    - AdminController
      - 'username' hides a field. - line: 86
      - 'path' hides a field. - line: 979
      - 'admin' hides a field. - line: 1215
      - 'name' hides a field. - line: 1491
    - AllAppointmentsController
      - 'receptionist' hides a field. - line: 27
    - AllMessages
      - 'sender' hides a field. - line: 19
      - 'type' hides a field. - line: 19
      - 'date' hides a field. - line: 19
      - 'subject' hides a field. - line: 19
      - 'id' hides a field. - line: 19
      - 'name' hides a field. - line: 19
      - 'string' hides a field. - line: 19
      - 'message' hides a field. - line: 19
      - 'string' hides a field. - line: 54
      - 'string' hides a field. - line: 64
      - 'string' hides a field. - line: 74
      - 'string' hides a field. - line: 84
      - 'string' hides a field. - line: 94
      - 'string' hides a field. - line: 104
      - 'string' hides a field. - line: 114
      - 'string' hides a field. - line: 124
    - AllMessagesController
      - 'newSysUser' hides a field. - line: 30
    - Appointment
      - 'startTime' hides a field. - line: 21

        - 'endTime' hides a field. - line: 29
    - Availability
        - 'id' hides a field. - line: 16
        - 'date' hides a field. - line: 16
        - 'time' hides a field. - line: 16
    - Bill
        - 'appointment' hides a field. - line: 26
        - 'billID' hides a field. - line: 26
        - 'hospital' hides a field. - line: 26
        - 'patientID' hides a field. - line: 26
        - 'doctor' hides a field. - line: 26
        - 'pharmacy' hides a field. - line: 26
        - 'laboratory' hides a field. - line: 26
        - 'bill' hides a field. - line: 26
        - 'date' hides a field. - line: 26
    - CashierController
        - 'username' hides a field. - line: 66
        - 'patientID' hides a field. - line: 298
        - 'billID' hides a field. - line: 449
        - 'name' hides a field. - line: 454
        - 'serviceFees' hides a field. - line: 526
        - 'name' hides a field. - line: 532
        - 'billID' hides a field. - line: 537
        - 'name' hides a field. - line: 854
    - CurrentUserSummaryController
        - 'sysUser' hides a field. - line: 28
    - DatabaseOperator
        - 'dbClassName' hides a field. - line: 106
    - DoctorController
        - 'username' hides a field. - line: 65
        - 'name' hides a field. - line: 1340
        - 'popup' hides a field. - line: 1526
        - 'popup' hides a field. - line: 1727
    - DoctorDetail
        - 'doctorName' hides a field. - line: 21
        - 'availability' hides a field. - line: 21
        - 'doctorID' hides a field. - line: 21
        - 'days' hides a field. - line: 21
        - 'area' hides a field. - line: 21
    - Drug
        - 'suppliers' hides a field. - line: 19
        - 'amount' hides a field. - line: 19
    - LabAssistantController
        - 'username' hides a field. - line: 67
        - 'name' hides a field. - line: 1315
    - LabReport
        - 'result' hides a field. - line: 16
        - 'constituent' hides a field. - line: 16
    - LabReportPreviewController
        - 'lab' hides a field. - line: 25
    - LoginController
        - 'username' hides a field. - line: 104
        - 'username' hides a field. - line: 131
        - 'username' hides a field. - line: 156

        - 'username' hides a field. - line: 179
        - 'username' hides a field. - line: 204
        - 'username' hides a field. - line: 232
    - LogoutController
        - 'user' hides a field. - line: 33
    - Message
        - 'sender' hides a field. - line: 16
        - 'message' hides a field. - line: 16
        - 'subject' hides a field. - line: 16
    - NewDoctorTimeSlotController
        - 'doc' hides a field. - line: 29
        - 'docC' hides a field. - line: 29
    - NewMessageController
        - 'newSysUser' hides a field. - line: 26
    - PharmacistController
        - 'username' hides a field. - line: 72
        - 'name' hides a field. - line: 529
        - 'name' hides a field. - line: 619
        - 'name' hides a field. - line: 969
    - PopoverController
        - 'text' hides a field. - line: 27
    - PopupAskController
        - 'refundC' hides a field. - line: 35
        - 'cashier' hides a field. - line: 35
        - 'label' hides a field. - line: 35
    - Prescription
        - 'prescID' hides a field. - line: 18
        - 'prescription' hides a field. - line: 18
        - 'doctor' hides a field. - line: 18
        - 'date' hides a field. - line: 18
    - PrescriptionListController
        - 'lab' hides a field. - line: 36
    - ReadMessageController
        - 'newSysUser' hides a field. - line: 32
        - 'message' hides a field. - line: 32
    - ReceptionistController
        - 'username' hides a field. - line: 60
        - 'name' hides a field. - line: 221
        - 'days' hides a field. - line: 228
        - 'name' hides a field. - line: 268
        - 'name' hides a field. - line: 750
        - 'name' hides a field. - line: 765
        - 'appDate' hides a field. - line: 914
        - 'name' hides a field. - line: 1240
        - 'appDate' hides a field. - line: 1457
        - 'days' hides a field. - line: 1710
    - Refund
        - 'date' hides a field. - line: 22
        - 'service' hides a field. - line: 22
        - 'patientID' hides a field. - line: 22
        - 'bill' hides a field. - line: 22
        - 'billID' hides a field. - line: 22
        - 'refund' hides a field. - line: 22
    - RefundController

        - 'cashier' hides a field. - line: 27
    - ReportsController
        - 'admin' hides a field. - line: 45
    - SettingsController
        - 'adminC' hides a field. - line: 38
        - 'admin' hides a field. - line: 38
        - 'result' hides a field. - line: 156
    - SysUserController
        - 'userID' hides a field. - line: 32
        - 'admin' hides a field. - line: 32
    - User
        - 'username' hides a field. - line: 78
        - 'username' hides a field. - line: 126
        - 'userType' hides a field. - line: 134
        - 'username' hides a field. - line: 142
        - 'username' hides a field. - line: 158
        - 'userID' hides a field. - line: 274
        - 'userID' hides a field. - line: 314
    - UserAccount
        - 'lastLogin' hides a field. - line: 24
        - 'online' hides a field. - line: 24
        - 'user' hides a field. - line: 24
    - UserAccountController
        - 'admin' hides a field. - line: 29
- Local Final Variable Name
    - DoctorController
        - Name 'Appointments' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 103
    - LabAssistantController
        - Name 'Appointments' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 527
- Local Variable Name
    - Admin
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 84
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 118
        - Name 'date_to_string' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 523
    - Cashier
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 354
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 388
    - CashierController
        - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 115
        - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 161
    - DatabaseOperator
        - Name 'TableColumnKeys' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 247
        - Name 'TableColumnNames' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 271
        - Name 'TableDataTypes' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 272
        - Name 'TableColumnNull' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 273
    - Doctor
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 98
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 127
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 156
    - DoctorController
        - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 171
        - Name 'DrugReactions' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 530
    - LabAssistant
        - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 70

- Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 98
- Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 127

- Pharmacist
  - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 893
  - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 921
  - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 950
- Receptionist
  - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 75
  - Name 'column_data' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 107
- ReceptionistController
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 113
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 157
  - Name 'tmp_day' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 929
- ReportsController
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 87
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 233
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 276
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 388
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 433
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 718
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 833
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 875
  - Name 'Month' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 918
- Validate
  - Name 'V' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 21

- Member Name
  - AdminController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 1348
  - CashierController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 713
  - DoctorController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 1516
  - LabAssistantController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 1158
  - PharmacistController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 813
  - PopupAskController
    - Name 'PopUpMessage' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 90
  - ReceptionistController
    - Name 'AllMessages' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 1341
  - SysUserController
    - Name 'NIC' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 55
    - Name 'DOB' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 56
- Method Name
  - DoctorController
    - Name 'MakeAvailabilityTable' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 1298
  - LabAssistant
    - Name 'UrineFullReport' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 353
    - Name 'LipidTest' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 407
    - Name 'BloodGroupingTest' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 460
    - Name 'RenalFunctionTest' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 587
    - Name 'SeriumCreatinePhosphokinaseTotal' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 637

        - Name 'SeriumCreatinePhosphokinase' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 687
    - PrescriptionListController
        - Name 'LoadPrescriptionInfo' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 54
    - Validate
        - Name 'NIC' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 15
- Package Name
    - AddNewDrugController.java
        - Name 'Pharmacist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Admin.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 7
    - AdminController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - AdminMessageController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - AllAppointmentsController.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Appointment.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - AppointmentSuccessController.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Availability.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Bill.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - BillPreviewController.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Cashier.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - CashierController.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Doctor.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - DoctorController.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - DoctorDetail.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 1
    - Drug.java
        - Name 'Pharmacist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - LabAssistant.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 5
    - LabAssistantController.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - LabReport.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - LabReportPreviewController.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Message.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - NewDoctorTimeSlotController.java

        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - NewUserController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 5
    - PatientAccountSuccessController.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Pharmacist.java
        - Name 'Pharmacist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - PharmacistController.java
        - Name 'Pharmacist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Popover2Controller.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - PopoverController.java
        - Name 'Doctor' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Prescription.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - PrescriptionListController.java
        - Name 'LabAssistant' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Receptionist.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - ReceptionistController.java
        - Name 'Receptionist' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - Refund.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - RefundController.java
        - Name 'Cashier' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - ReportsController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - SettingsController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - SysUserController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - UserAccount.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 5
    - UserAccountController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 0
    - UserOptionPopOverController.java
        - Name 'Admin' must match pattern '^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$'. - line: 5
- Parameter Assignment
    - Doctor
        - Assignment of parameter 'tests' is not allowed. - line: 749
        - Assignment of parameter 'tests' is not allowed. - line: 750
    - LabAssistant
        - Assignment of parameter 'month' is not allowed. - line: 284
    - Receptionist
        - Assignment of parameter 'day' is not allowed. - line: 341
- Parameter Name
    - BillPreviewController
        - Name 'VAT' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 48
    - DatabaseOperator
        - Name 'ColumnName' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 346
        - Name 'ColumnName' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 364
    - LabAssistant

        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 353
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 407
        - Name 'app_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 460
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 528
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 587
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 637
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 687
        - Name 'appointment_id' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 745
    - Pharmacist
        - Name 'Name' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 580
    - Receptionist
        - Name 'ConsultationArea' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 887
- Static Variable Name
    - DatabaseOperator
        - Name 'CONNECTION' must match pattern '^[a-z][a-zA-Z0-9]*$'. - line: 97