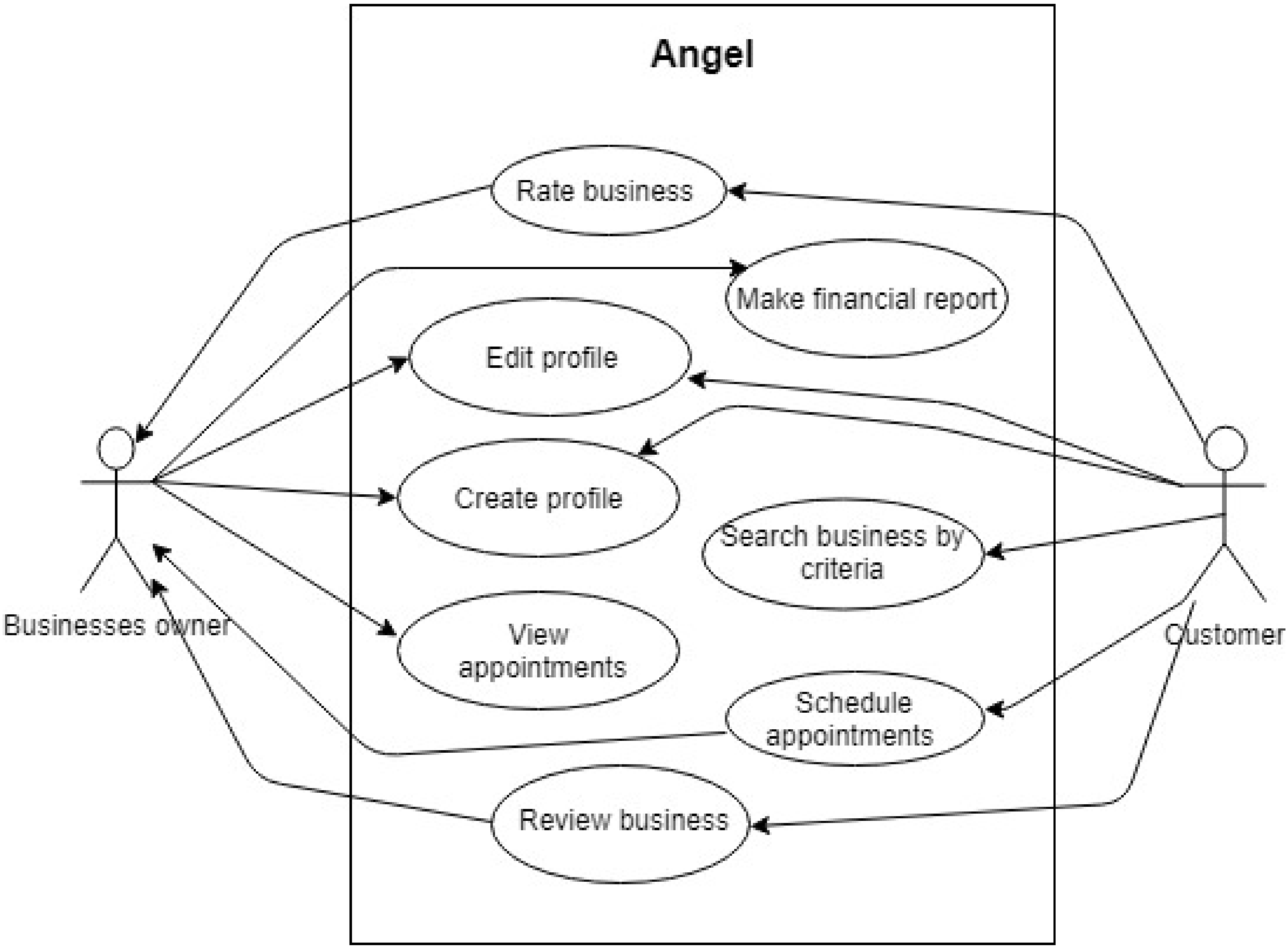
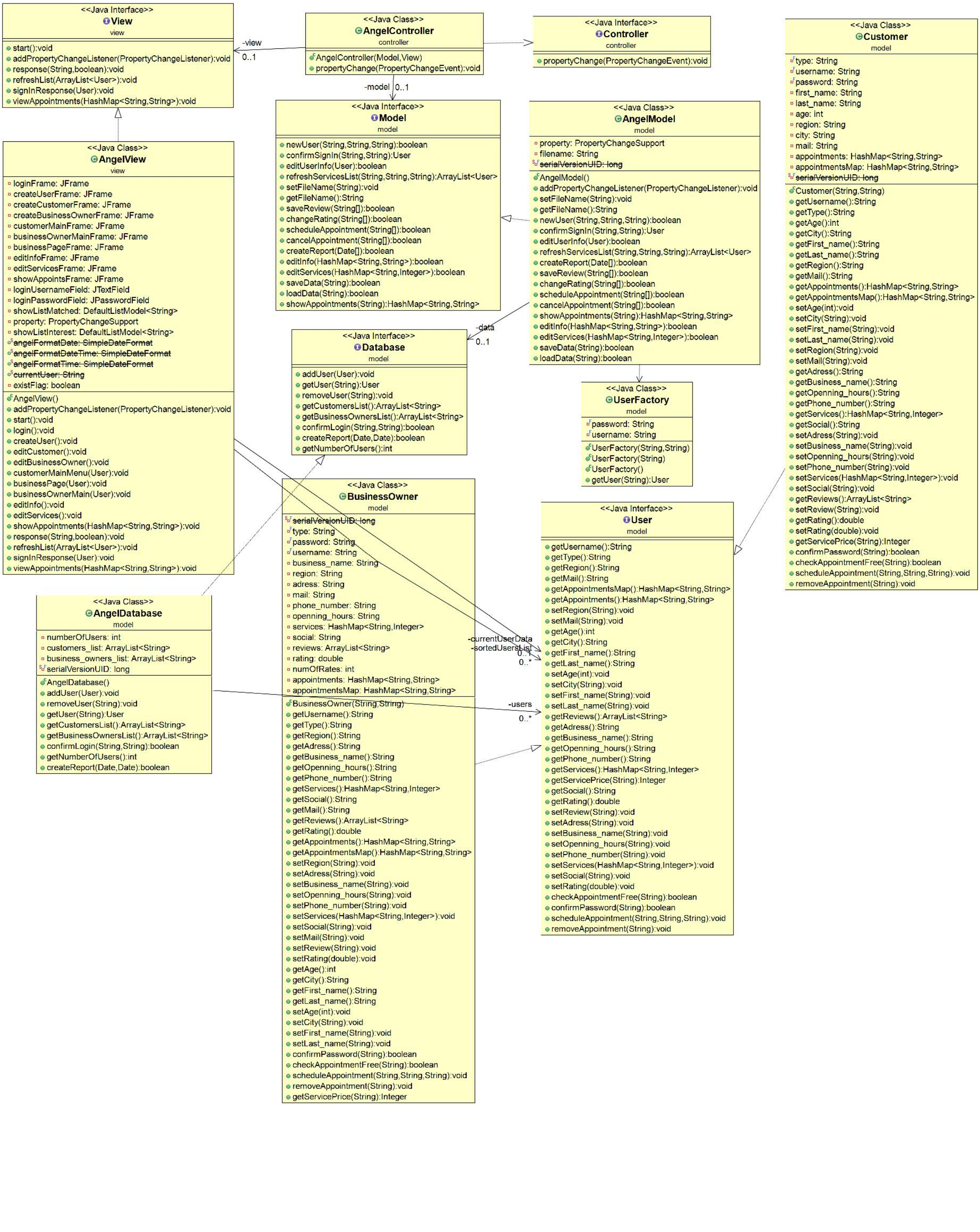
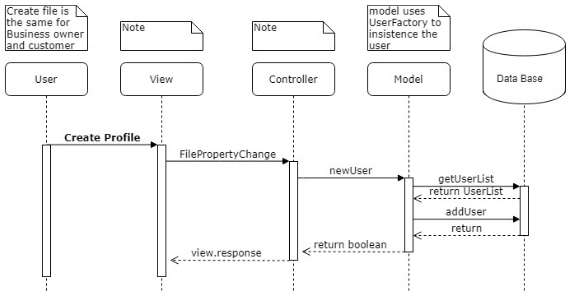
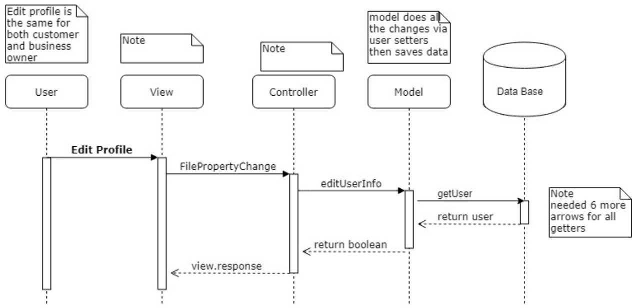
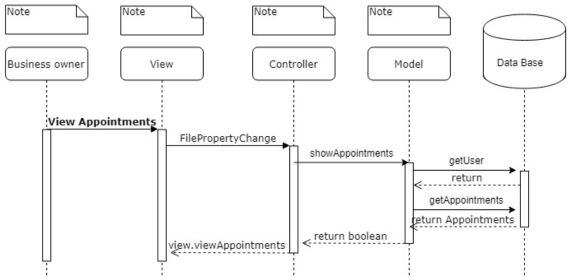
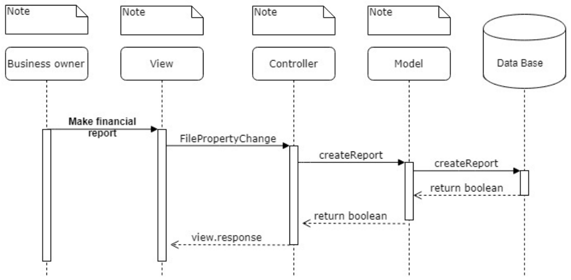

Angel
Rate business
Make financial report
Edit profile
Create profile
Search business by criteria
View appointments
Schedule appointments
Review business
Businesses owner
Customer

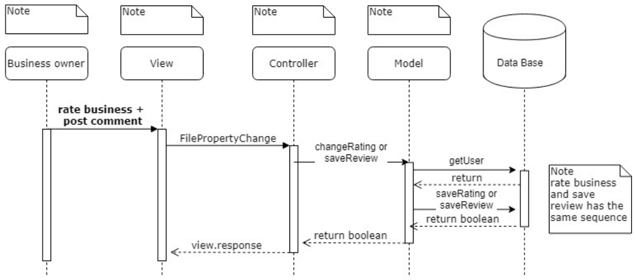

<<Java Interface>>
View
view
start():void
addPropertyChangeListener(PropertyChangeListener):void
response(String,boolean):void
refreshList(ArrayList<User>):void
signInResponse(User):void
viewAppointments(HashMap<String,String>):void
-view
0..1
<<Java Class>>
AngelController
controller
AngelController(Model,View)
propertyChange(PropertyChangeEvent):void
<<Java Interface>>
Controller
controller
propertyChange(PropertyChangeEvent):void
-model
0..1
<<Java Interface>>
Model
model
newUser(String,String,String):boolean
confirmSignIn(String,String):User
editUserInfo(User):boolean
refreshServicesList(String,String,String):ArrayList<User>
setFileName(String):void
getFileName():String
saveReview(String[]):boolean
changeRating(String[]):boolean
scheduleAppointment(String[]):boolean
cancelAppointment(String[]):boolean
createReport(Date[]):boolean
editInfo(HashMap<String,String>):boolean
editServices(HashMap<String,Integer>):boolean
saveData(String):boolean
loadData(String):boolean
showAppointments(String):HashMap<String,String>
<<Java Class>>
AngelModel
model
property: PropertyChangeSupport
filename: String
~~serialVersionUID: long~~
AngelModel()
addPropertyChangeListener(PropertyChangeListener):void
setFileName(String):void
getFileName():String
newUser(String,String,String):boolean
confirmSignIn(String,String):User
editUserInfo(User):boolean
refreshServicesList(String,String,String):ArrayList<User>
createReport(Date[]):boolean
saveReview(String[]):boolean
changeRating(String[]):boolean
scheduleAppointment(String[]):boolean
cancelAppointment(String[]):boolean
showAppointments(String):HashMap<String,String>
editInfo(HashMap<String,String>):boolean
editServices(HashMap<String,Integer>):boolean
saveData(String):boolean
loadData(String):boolean
-data
0..1
<<Java Class>>
Customer
model
type: String
username: String
password: String
first_name: String
last_name: String
age: int
region: String
city: String
mail: String
appointments: HashMap<String,String>
appointmentsMap: HashMap<String,String>
~~serialVersionUID: long~~
Customer(String,String)
getUsername():String
getType():String
getAge():int
getCity():String
getFirst_name():String
getLast_name():String
getRegion():String
getMail():String
getAppointments():HashMap<String,String>
getAppointmentsMap():HashMap<String,String>
setAge(int):void
setCity(String):void
setFirst_name(String):void
setLast_name(String):void
setRegion(String):void
setMail(String):void
getAdress():String
getBusiness_name():String
getOpenning_hours():String
getPhone_number():String
getServices():HashMap<String,Integer>
getSocial():String
setAdress(String):void
setBusiness_name(String):void
setOpenning_hours(String):void
setPhone_number(String):void
setServices(HashMap<String,Integer>):void
setSocial(String):void
getReviews():ArrayList<String>
setReview(String):void
getRating():double
setRating(double):void
getServicePrice(String):Integer
confirmPassword(String):boolean
checkAppointmentFree(String):boolean
scheduleAppointment(String,String,String):void
removeAppointment(String):void
<<Java Class>>
AngelView
view
loginFrame: JFrame
createUserFrame: JFrame
createCustomerFrame: JFrame
createBusinessOwnerFrame: JFrame
customerMainFrame: JFrame
businessOwnerMainFrame: JFrame
businessPageFrame: JFrame
editInfoFrame: JFrame
editServicesFrame: JFrame
showAppointsFrame: JFrame
loginUsernameField: JTextField
loginPasswordField: JPasswordField
showListMatched: DefaultListModel<String>
property: PropertyChangeSupport
showListInterest: DefaultListModel<String>
~~angelFormatDate: SimpleDateFormat~~
~~angelFormatDateTime: SimpleDateFormat~~
~~angelFormatTime: SimpleDateFormat~~
~~currentUser: String~~
existFlag: boolean
AngelView()
addPropertyChangeListener(PropertyChangeListener):void
start():void
login():void
createUser():void
editCustomer():void
editBusinessOwner():void
customerMainMenu(User):void
businessPage(User):void
businessOwnerMain(User):void
editInfo():void
editServices():void
showAppointments(HashMap<String,String>):void
response(String,boolean):void
refreshList(ArrayList<User>):void
signInResponse(User):void
viewAppointments(HashMap<String,String>):void
<<Java Interface>>
Database
model
addUser(User):void
getUser(String):User
removeUser(String):void
getCustomersList():ArrayList<String>
getBusinessOwnersList():ArrayList<String>
confirmLogin(String,String):boolean
createReport(Date,Date):boolean
getNumberOfUsers():int
<<Java Class>>
UserFactory
model
password: String
username: String
UserFactory(String,String)
UserFactory(String)
UserFactory()
getUser(String):User
<<Java Class>>
BusinessOwner
model
~~serialVersionUID: long~~
type: String
password: String
username: String
business_name: String
region: String
adress: String
mail: String
phone_number: String
openning_hours: String
services: HashMap<String,Integer>
social: String
reviews: ArrayList<String>
rating: double
numOfRates: int
appointments: HashMap<String,String>
appointmentsMap: HashMap<String,String>
BusinessOwner(String,String)
getUsername():String
getType():String
getRegion():String
getAdress():String
getBusiness_name():String
getOpenning_hours():String
getPhone_number():String
getServices():HashMap<String,Integer>
getSocial():String
getMail():String
getReviews():ArrayList<String>
getRating():double
getAppointments():HashMap<String,String>
getAppointmentsMap():HashMap<String,String>
setRegion(String):void
setAdress(String):void
setBusiness_name(String):void
setOpenning_hours(String):void
setPhone_number(String):void
setServices(HashMap<String,Integer>):void
setSocial(String):void
setMail(String):void
setReview(String):void
setRating(double):void
getAge():int
getCity():String
getFirst_name():String
getLast_name():String
setAge(int):void
setCity(String):void
setFirst_name(String):void
setLast_name(String):void
confirmPassword(String):boolean
checkAppointmentFree(String):boolean
scheduleAppointment(String,String,String):void
removeAppointment(String):void
getServicePrice(String):Integer
<<Java Interface>>
User
model
getUsername():String
getType():String
getRegion():String
getMail():String
getAppointmentsMap():HashMap<String,String>
getAppointments():HashMap<String,String>
setRegion(String):void
setMail(String):void
getAge():int
getCity():String
getFirst_name():String
getLast_name():String
setAge(int):void
setCity(String):void
setFirst_name(String):void
setLast_name(String):void
getReviews():ArrayList<String>
getAdress():String
getBusiness_name():String
getOpenning_hours():String
getPhone_number():String
getServices():HashMap<String,Integer>
getServicePrice(String):Integer
getSocial():String
getRating():double
setReview(String):void
setAdress(String):void
setBusiness_name(String):void
setOpenning_hours(String):void
setPhone_number(String):void
setServices(HashMap<String,Integer>):void
setSocial(String):void
setRating(double):void
checkAppointmentFree(String):boolean
confirmPassword(String):boolean
scheduleAppointment(String,String,String):void
removeAppointment(String):void
-currentUserData
-sortedUsersList
0..1
0..*
-users
0..*
<<Java Class>>
AngelDatabase
model
numberOfUsers: int
customers_list: ArrayList<String>
business_owners_list: ArrayList<String>
serialVersionUID: long
AngelDatabase()
addUser(User):void
removeUser(String):void
getUser(String):User
getCustomersList():ArrayList<String>
getBusinessOwnersList():ArrayList<String>
confirmLogin(String,String):boolean
getNumberOfUsers():int
createReport(Date,Date):boolean

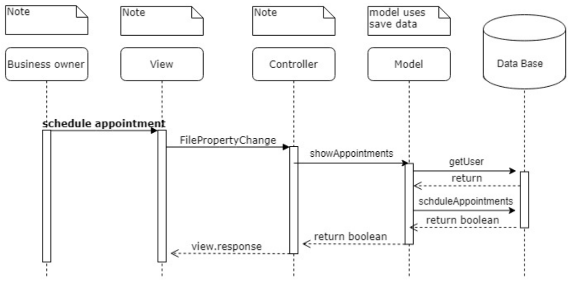

Create file is the same for Business owner and customer
Note
Note
model uses UserFactory to insistence the user
User
View
Controller
Model
Data Base
Create Profile
FilePropertyChange
newUser
getUserList
return UserList
addUser
return
return boolean
view.response

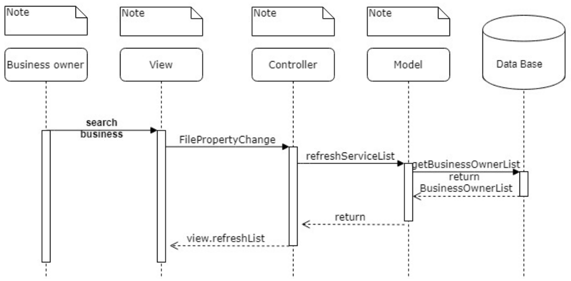

Edit profile is the same for both customer and business owner
Note
Note
model does all the changes via user setters then saves data
User
View
Controller
Model
Data Base
Edit Profile
FilePropertyChange
editUserInfo
getUser
return user
Note needed 6 more arrows for all getters
return boolean
view.response


Note
Note
Note
Note
Business owner
View
Controller
Model
Data Base
View Appointments
FilePropertyChange
showAppointments
getUser
return
getAppointments
return Appointments
return boolean
view.viewAppointments


Note
Note
Note
Note
Business owner
View
Controller
Model
Data Base
Make financial report
FilePropertyChange
createReport
createReport
return boolean
return boolean
view.response


Note
Note
Note
Note
Business owner
View
Controller
Model
Data Base
rate business + post comment
FilePropertyChange
changeRating or saveReview
getUser
return
saveRating or saveReview
return boolean
Note rate business and save review has the same sequence
return boolean
view.response


Note
Note
Note
model uses save data
Business owner
View
Controller
Model
Data Base
schedule appointment
FilePropertyChange
showAppointments
getUser
return
schduleAppointments
return boolean
return boolean
view.response


Note
Note
Note
Note
Business owner
View
Controller
Model
Data Base
search business
FilePropertyChange
refreshServiceList
getBusinessOwnerList
return BusinessOwnerList
return
view.refreshList

Architectural pattern

The Architectural we will focus on is the Model View Controller.

The 3-part architecture in the system is used to help us control the beat in our system.

The strategy:

The view is an object that is configured with strategy, the controller provides with strategy, the view is concerned only with the visual aspects of the application, and delegated to the controller for any decisions about the interface.The model implements the observer pattern to keep interested objects updated when state changes occur.Using the observer pattern keep the model completely independent of the views and controller. In our design, the view is a consists of a nested set of windows, panels and buttons, when the controller tells the view to update via "`propertyChange`" function, it only has to tell the top view component (view.response()), and composite takes care of the rest, furthermore the controller has model composite, and the flow can go in several directions, the controller can execute model to save changes and update data (composite) in our system. The view can notify the controller that something that something has accrued via "`PropertyChangeListener` property" that fires up "`firePropertyChange`".