UNITED REAL ESTATE ONLINE

SOFTWARE DESIGN DOCUMENT

Team 3
Ryan Gadhi
Asaad AlGhamdi
Mohammed AlJarbou

Table of Contents

Document Component	Weight	Grade
Organization	5%	
Introduction	5%	
Software Architecture	20%	
Structure Design	40%	
Behavior Design	20%	
	Total	

Introduction

Product Introduction

This software design document describes fully the United Real Estate Online system that is agreed upon on signed contract by United Real Estate Company. This document will describe the classes and main components that is expected of the system to have and their interaction, and any external interfaces which interact with the system and users, expected classes attributes and classes associations. The United Real Estate Online system is a three-tier repository system whose main functionalities and responsibilities are to facilitate, manage and automate the sales, investments, advertisement and customers' data storage through multiple user interfaces.

Purpose

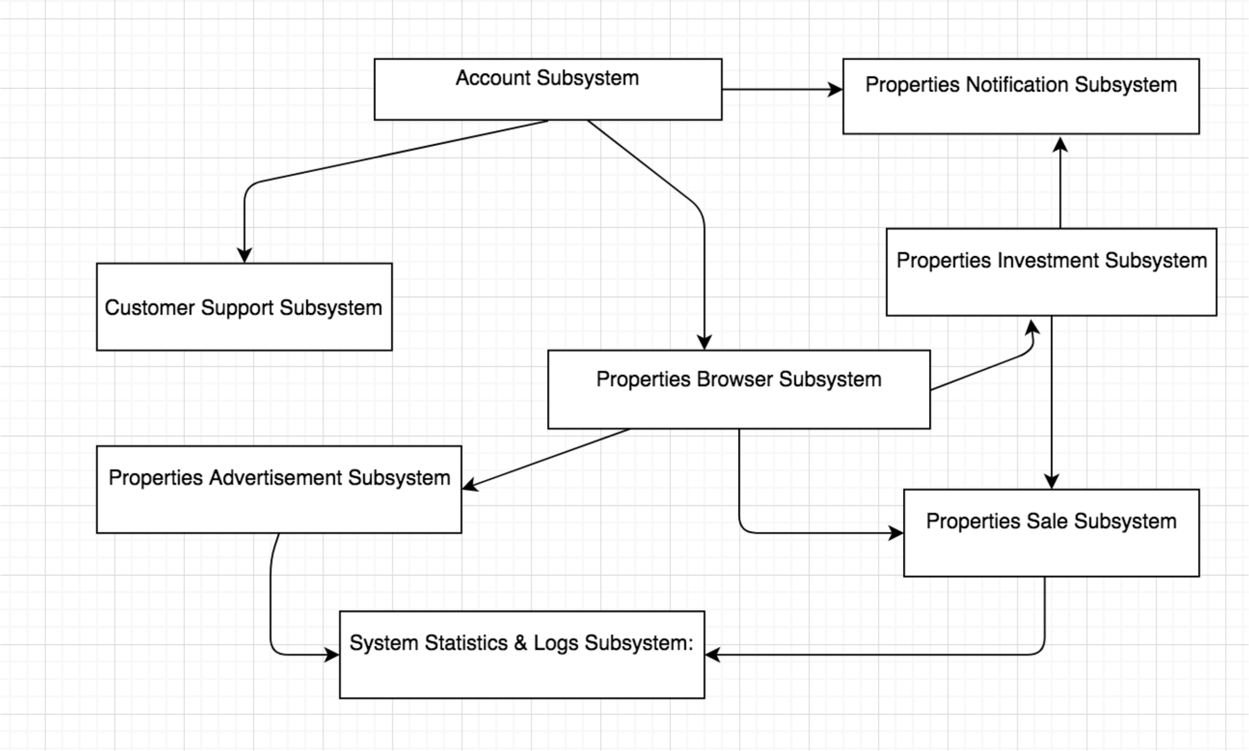
The purpose of this document is to illustrate the classes and main components of the United Real Estate Online system and what are their expected attributes, behavior and associations with each other.

The main objective of this document is to create a common space between the stakeholders, requirement engineers and the programmers that will implement the system, which shows a clear guideline on how to implement system and illustrate the software structure and architecture of the system, its components and classes. The advantage of this document is to clear any confusion or misunderstanding between requirement phase and implementation phase, leaving no space for requirement leakage or ambiguity.

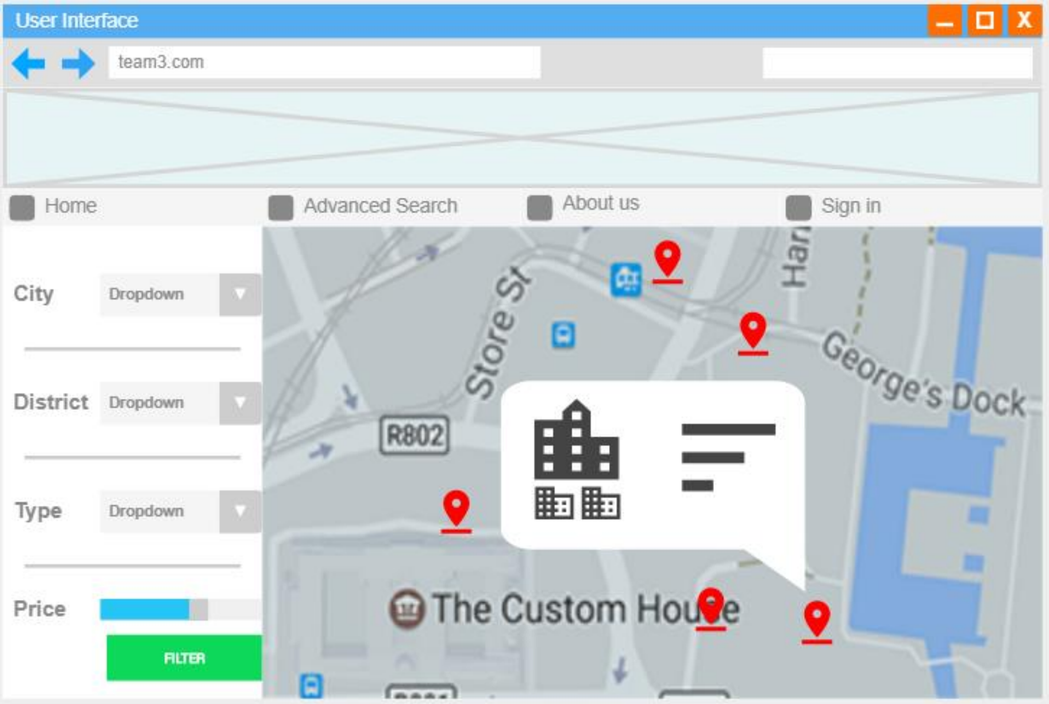
Overview

The remainder of the document will illustrate the United Real Estate Online software architecture and go into details about the reason such an architecture was chosen. Additionally, the user interface design of the system will be illustrated, which will describe the major functionalities of the software. Then, a structural description of the software system classes will be given, describing the main classes, attributes and methods of the United Real Estate System. Last but not least, a behavioral description through state diagrams of the critical complex systems of the software will be given for each complex subsystem existing within the software leaving no room for ambiguity.

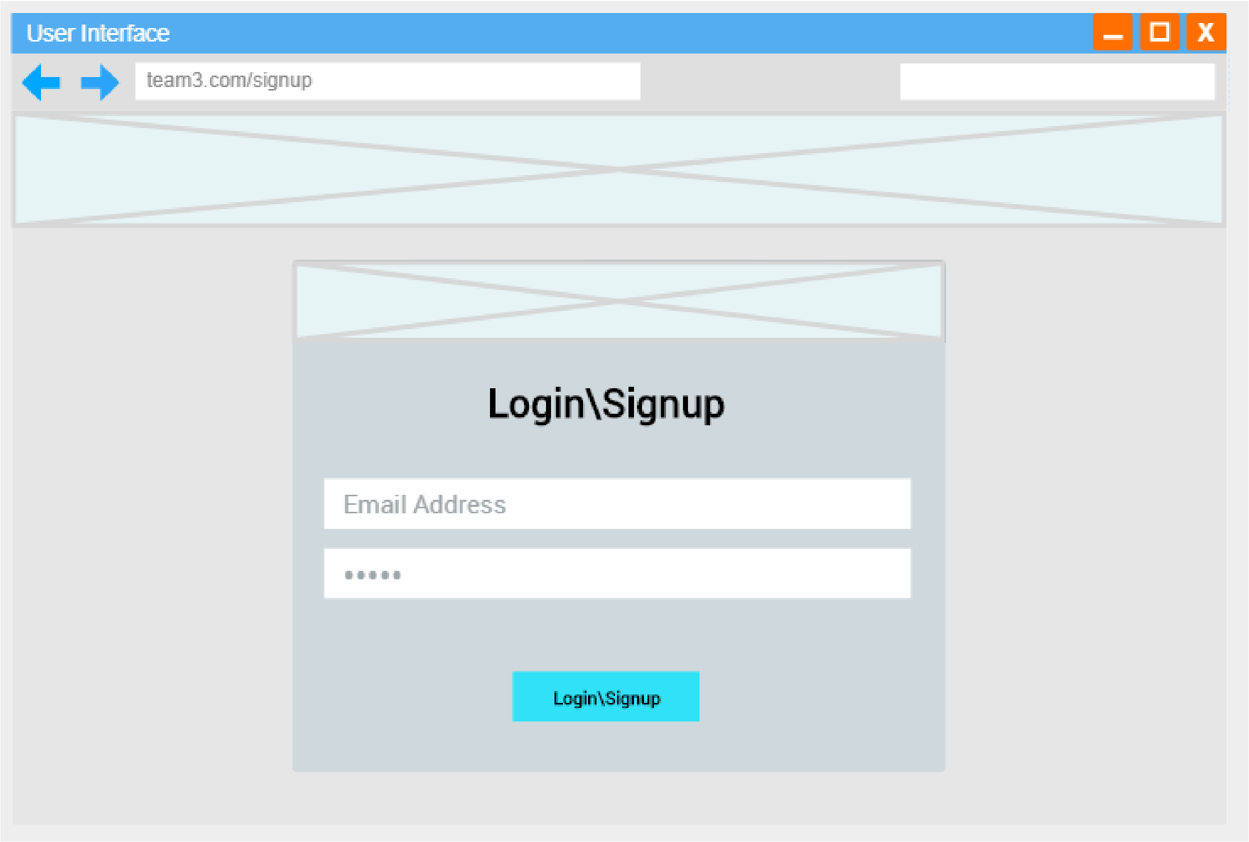
Software Architecture


Account Subsystem
Properties Notification Subsystem
Properties Investment Subsystem
Customer Support Subsystem
Properties Browser Subsystem
Properties Advertisement Subsystem
Properties Sale Subsystem
System Statistics & Logs Subsystem:

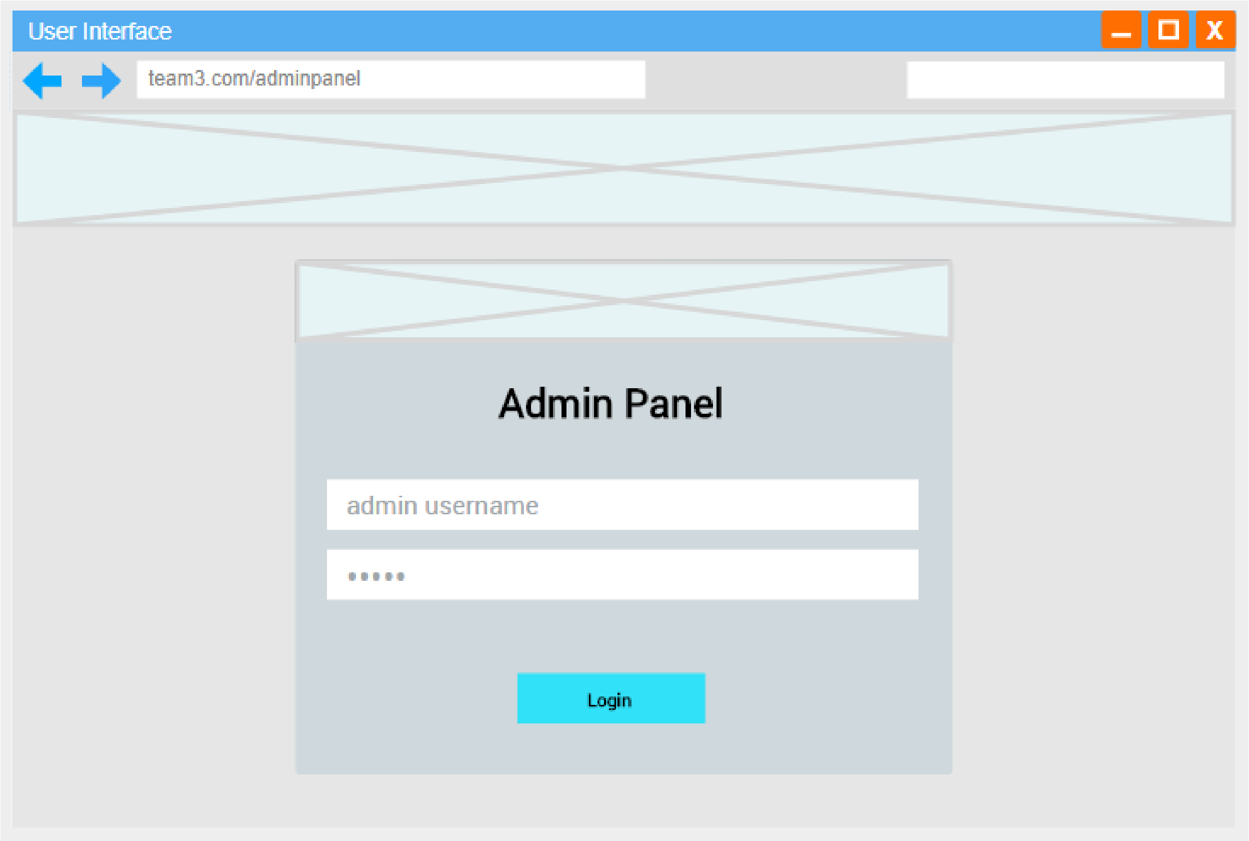
User Interface Design



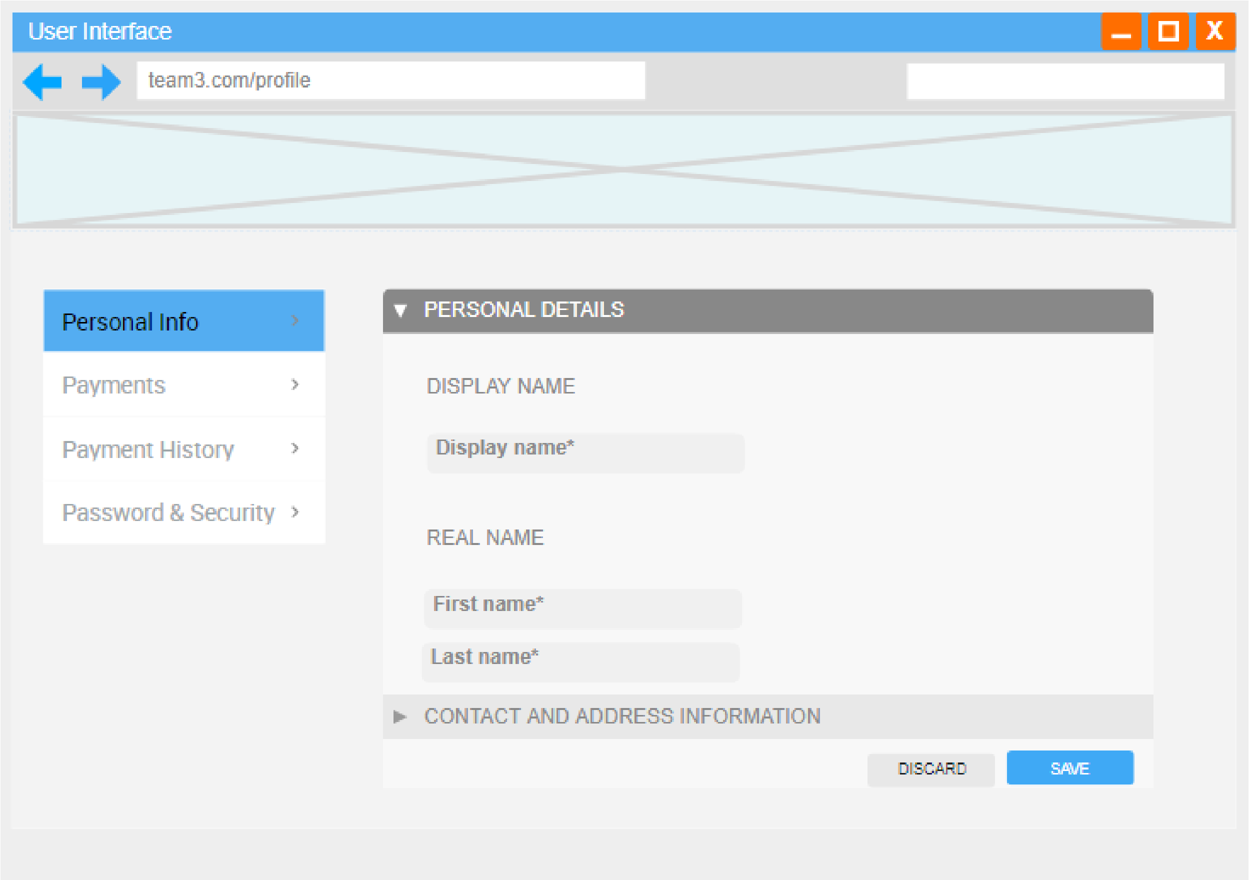
Home Page


User Interface
team3.com/signup
Login\Signup
Email Address
•••••
Login\Signup

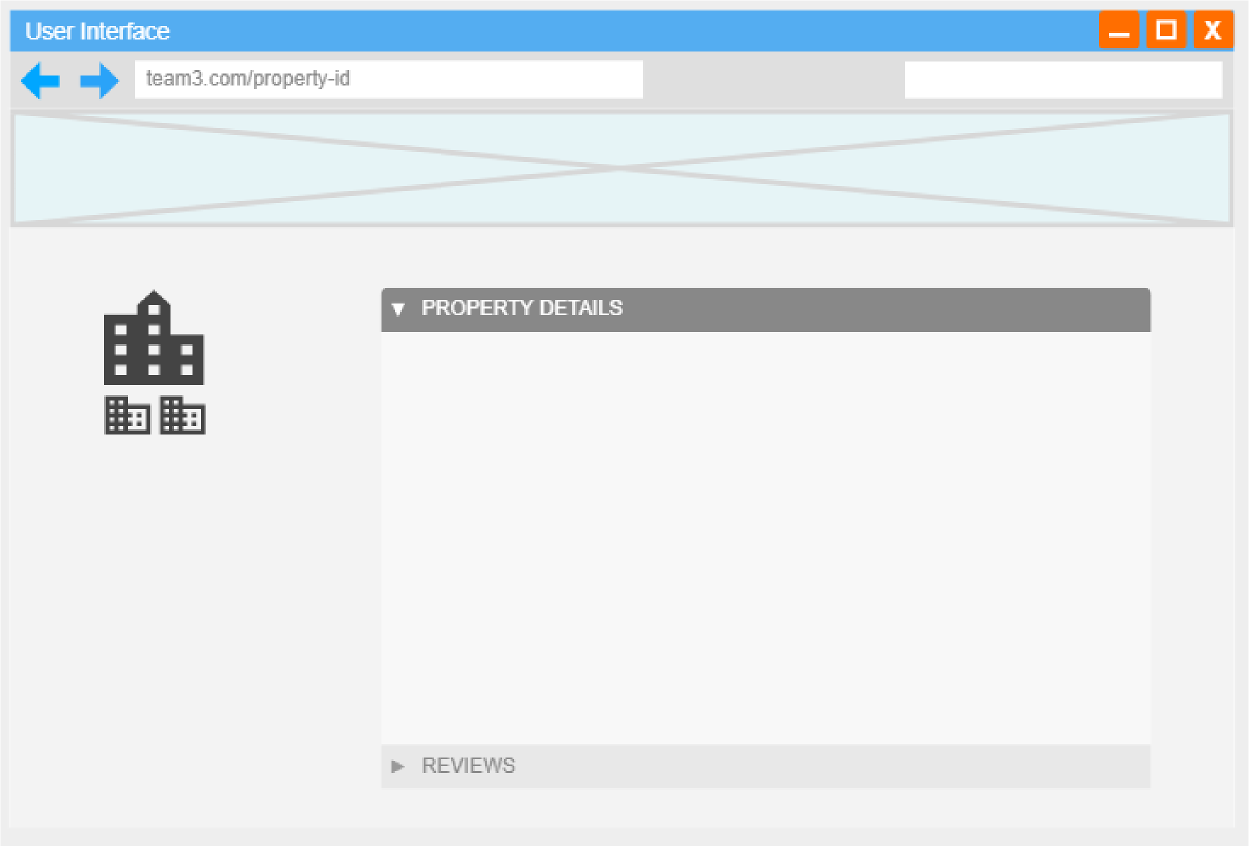
Login


User Interface
team3.com/adminpanel
Admin Panel
admin username
•••••
Login

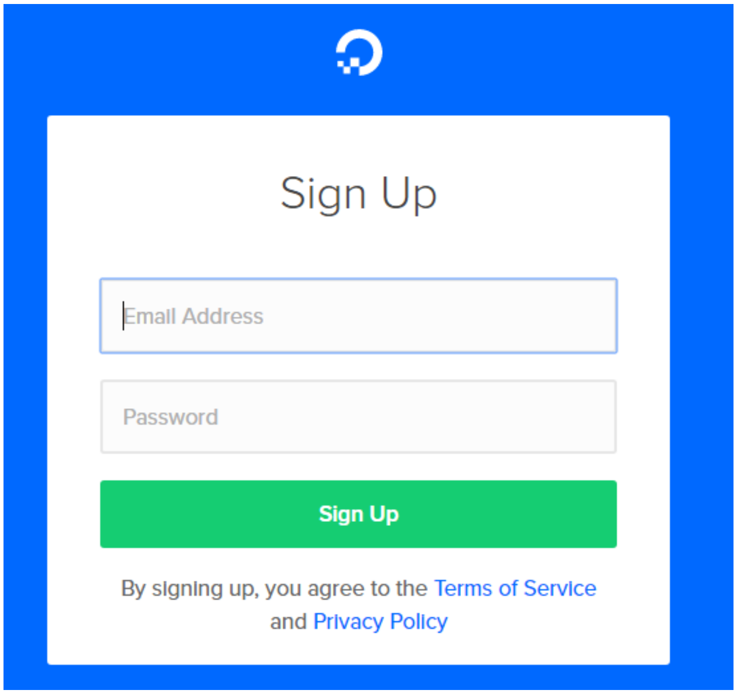
Admin Panel



User Profile



Property Profile

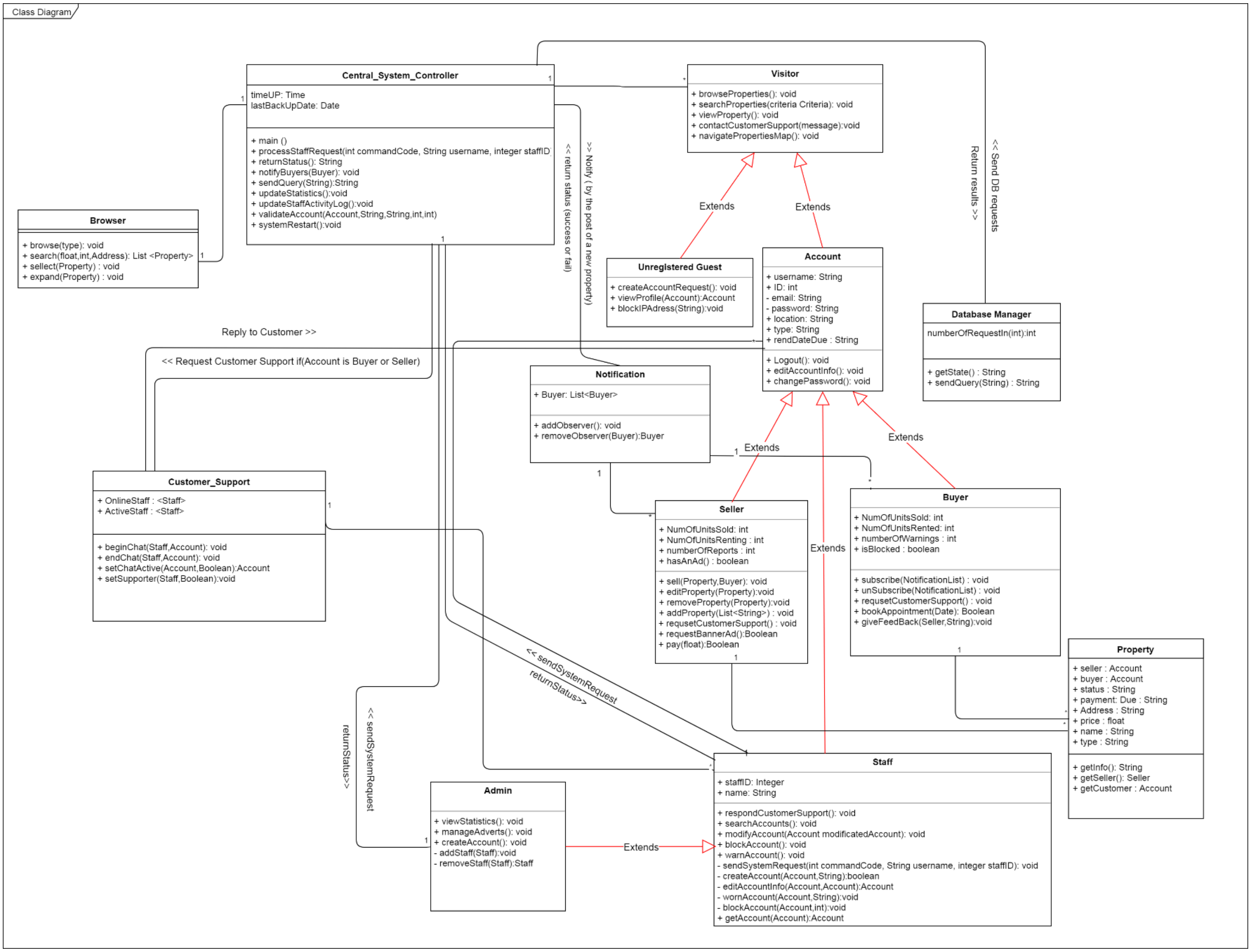


By signing up, you agree to the Terms of Service and Privacy Policy

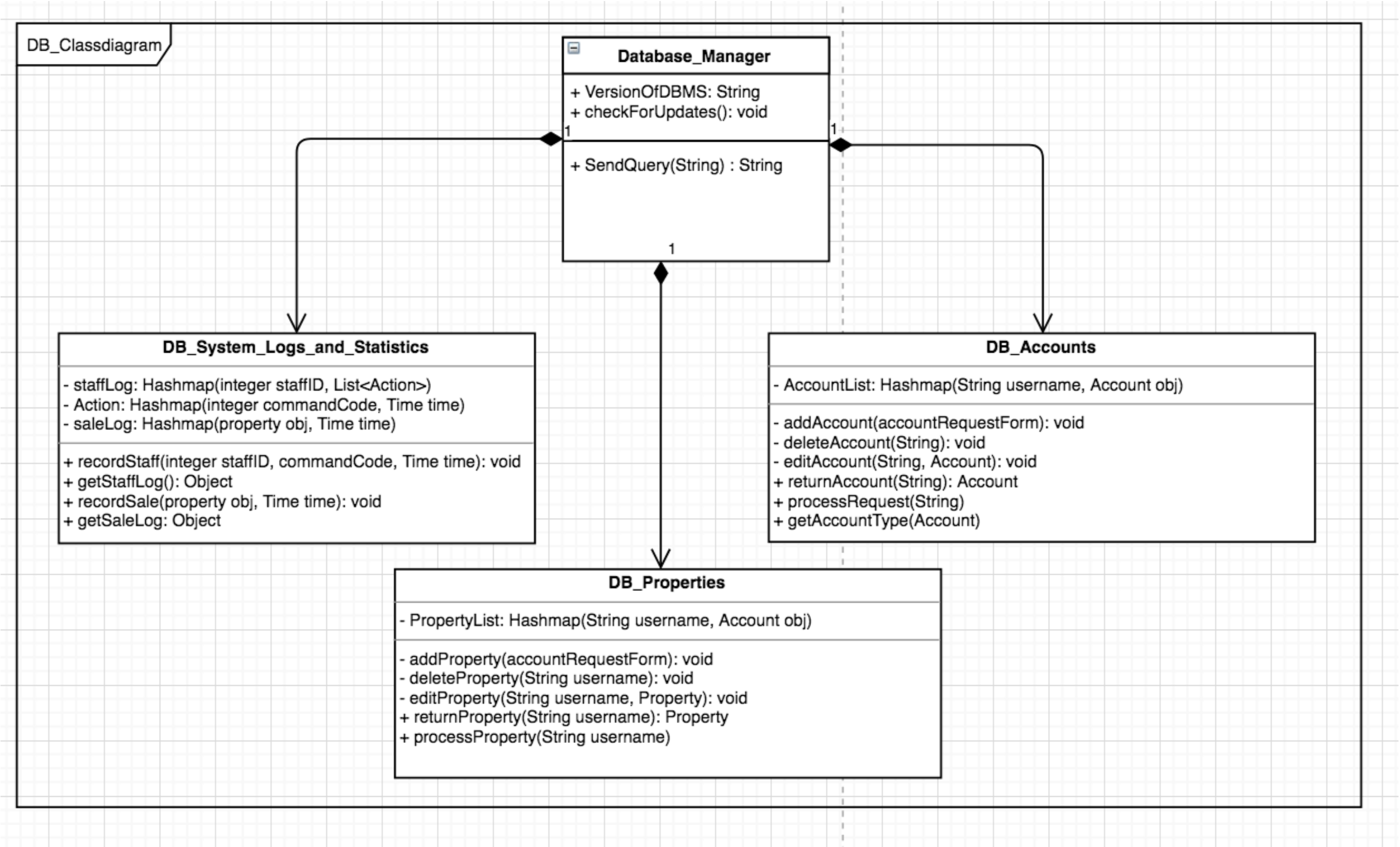
Sign-up Form

Class Diagrams

➢ Main Diagram


Class Diagram
Central_System_Controller
timeUP: Time
lastBackUpDate: Date
+ main ()
+ processStaffRequest(int commandCode, String username, integer staffID
+ returnStatus(): String
+ notifyBuyers(Buyer): void
+ sendQuery(String):String
+ updateStatistics():void
+ updateStaffActivityLog():void
+ validateAccount(Account,String,String,int,int)
+ systemRestart():void
Browser
+ browse(type): void
+ search(float,int,Address): List <Property>
+ sellect(Property) : void
+ expand(Property) : void
Visitor
+ browseProperties(): void
+ searchProperties(criteria Criteria): void
+ viewProperty(): void
+ contactCustomerSupport(message):void
+ navigatePropertiesMap(): void
Extends
Extends
Unregistered Guest
+ createAccountRequest(): void
+ viewProfile(Account):Account
+ blockIPAdress(String):void
Account
+ username: String
+ ID: int
- email: String
- password: String
+ location: String
+ type: String
+ rendDateDue : String
+ Logout(): void
+ editAccountInfo(): void
+ changePassword(): void
Database Manager
numberOfRequestIn(int):int
+ getState() : String
+ sendQuery(String) : String
<< Send DB requests
Return results >>
<< return status (success or fail)
>> Notify (by the post of a new property)
Reply to Customer >>
<< Request Customer Support if(Account is Buyer or Seller)
Notification
+ Buyer: List<Buyer>
+ addObserver(): void
+ removeObserver(Buyer):Buyer
Extends
Extends
Extends
Customer_Support
+ OnlineStaff : <Staff>
+ ActiveStaff : <Staff>
+ beginChat(Staff,Account): void
+ endChat(Staff,Account): void
+ setChatActive(Account,Boolean):Account
+ setSupporter(Staff,Boolean):void
Seller
+ NumOfUnitsSold: int
+ NumOfUnitsRenting : int
+ numberOfReports : int
+ hasAnAd() : boolean
+ sell(Property,Buyer): void
+ editProperty(Property):void
+ removeProperty(Property):void
+ addProperty(List<String>) : void
+ requsetCustomerSupport() : void
+ requestBannerAd():Boolean
+ pay(float):Boolean
Buyer
+ NumOfUnitsSold: int
+ NumOfUnitsRented: int
+ numberOfWarnings : int
+ isBlocked : boolean
+ subscribe(NotificationList) : void
+ unSubscribe(NotificationList) : void
+ requsetCustomerSupport() : void
+ bookAppointment(Date): Boolean
+ giveFeedBack(Seller,String):void
Property
+ seller : Account
+ buyer : Account
+ status : String
+ payment: Due : String
+ Address : String
+ price : float
+ name : String
+ type : String
+ getInfo(): String
+ getSeller(): Seller
+ getCustomer : Account
<< sendSystemRequest
returnStatus>>
<< sendSystemRequest
returnStatus>>
Staff
+ staffID: Integer
+ name: String
+ respondCustomerSupport(): void
+ searchAccounts(): void
+ modifyAccount(Account modificatedAccount): void
+ blockAccount(): void
+ warnAccount(): void
- sendSystemRequest(int commandCode, String username, integer staffID): void
- createAccount(Account,String):boolean
- editAccountInfo(Account,Account):Account
- wornAccount(Account,String):void
- blockAccount(Account,int):void
+ getAccount(Account):Account
Admin
+ viewStatistics(): void
+ manageAdverts(): void
+ createAccount(): void
- addStaff(Staff):void
- removeStaff(Staff):Staff
Extends

- **Database Diagram**


DB_Classdiagram
Database_Manager
+ VersionOfDBMS: String
+ checkForUpdates(): void
+ SendQuery(String) : String
1
1
1
DB_System_Logs_and_Statistics
- staffLog: Hashmap(integer staffID, List<Action>)
- Action: Hashmap(integer commandCode, Time time)
- saleLog: Hashmap(property obj, Time time)
+ recordStaff(integer staffID, commandCode, Time time): void
+ getStaffLog(): Object
+ recordSale(property obj, Time time): void
+ getSaleLog: Object
DB_Accounts
- AccountList: Hashmap(String username, Account obj)
- addAccount(accountRequestForm): void
- deleteAccount(String): void
- editAccount(String, Account): void
+ returnAccount(String): Account
+ processRequest(String)
+ getAccountType(Account)
DB_Properties
- PropertyList: Hashmap(String username, Account obj)
- addProperty(accountRequestForm): void
- deleteProperty(String username): void
- editProperty(String username, Property): void
+ returnProperty(String username): Property
+ processProperty(String username)

Class diagram classes descriptions

Central_System_Controller	
Attributes	
timeUP: Time	The time count since the system started working
lastBackUpDate: Date	The date of the last back up of the system
Methods	
➢ **main ()**	The method that start the program
➢ **processStaffRequest(int commandCode, String username, integer staffID)**	Processing the requests gotten from the staff
➢ **returnStatus(): String**	Return the status , will be overridden
➢ **notifyBuyers(Buyer): void**	Notify buyers with a new item from the seller they subscribed to
➢ **sendQuery(String):String**	Send direct queries to the database
➢ **updateStatistics():void**	Fetch the new content of the database and update the statistics class
➢ **updateStaffActivityLog():void**	A direct way of updating the activity log of the staff
➢ **validateAccount(Account** ➢ **,String,String,int,int)**	Compares the provided account with the list of accounts in the database
➢ **systemRestart():void**	Requested a system restart

Visitor

Methods

- **browseProperties(): void**	displays a list of properties
- **searchProperties(criteria Criteria): void**	Takes a criteria of a search and , invoke the fetching
- **viewProperty(): void**	Fetches the info of the property from the database and displays it.
- **contactCustomerSupport(message):void**	Request a customer support , with a field for describing the issue
- **navigatePropertiesMap(): void**	Displays a map for property navigation

Account

Attributes

- **username: String**	The user name
- **ID: int**	Identification number
- **email: String**	Email address
- **password: String**	Password
- **location: String**	Location of the user
- **type: String**	Types either staff , buyer or seller
- **rendDateDue : String**	This describes the due date of the property payment

Methods

Logout(): void	Logout option
editAccountInfo(): void	Fetches the data from the DB and resends the updates
changePassword(): void	

Seller	
Attributes	
➢ NumOfUnitsSold: int	.
➢ NumOfUnitsRenting : int	.
	Fetches the data from the DB and resends the updates
➢ numberOfReports : int	.
➢ hasAnAd() : boolean	Checks if the seller has an id and returns boolean
Methods	
➢ sell(Property,Buyer): void	Allows the seller to take the selling action
➢ editProperty(Property):void	
➢ removeProperty(Property):void	Fetches the data from the DB and resends the updates
➢ addProperty(List<String>) : void	
➢ requsetCustomerSupport() : void	Shows an option to request customer support
➢ requestBannerAd():Boolean	Sends request to the staff regarding adding of the advertisements
➢ pay(float):Boolean	Handles the payment from the seller to the advertisement

Unregistered Guest	
Methods	
createAccountRequest(): void	Sends request to the controller to make a new account
viewProfile(Account):Account	Fetches the data from the DB and displays the account info
blockIPAdress(String):void	Blocks the user IP if they are trying to login multiple times with wrong password or username

Admin	
Methods	
viewStatistics(): void	Fetches the statistics from the database and displays it
manageAdverts(): void	Allows the admin to put the chosen ads on specific chosen places on the software
createAccount(): void	
addStaff(Staff):void	Creates a new staff account
removeStaff(Staff):Staff	Deletes a staff account

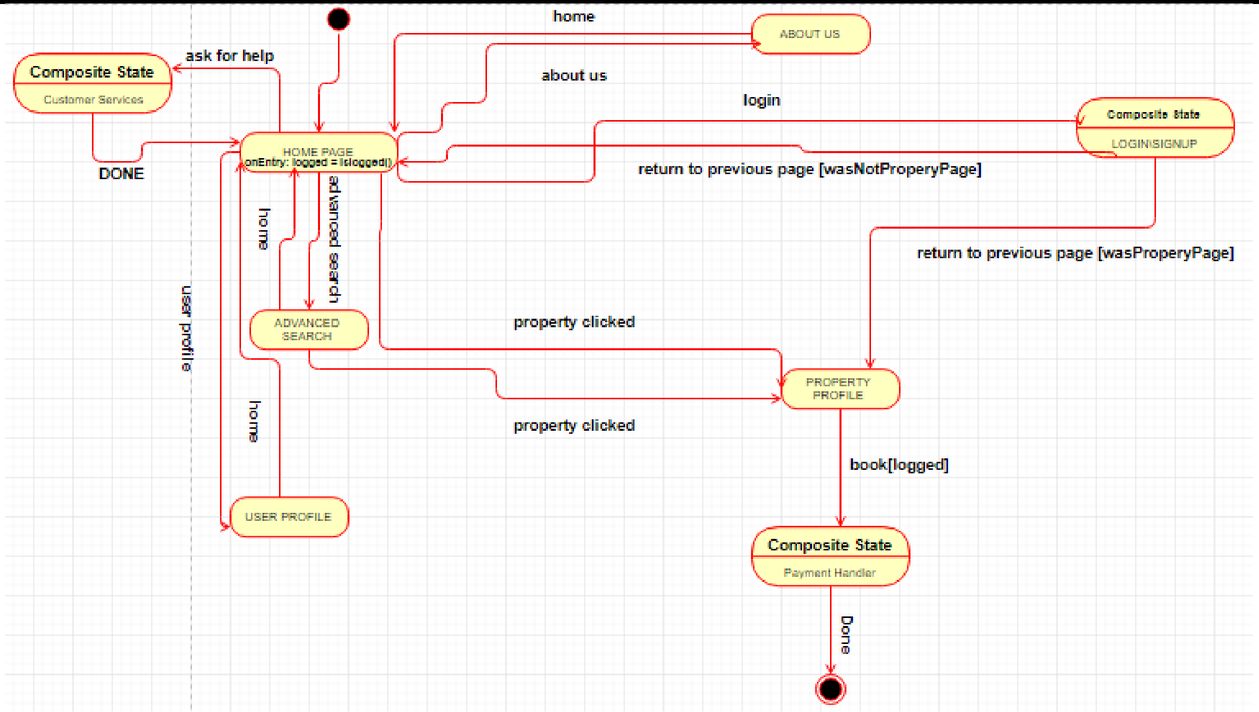
Customer Support	
Attributes	
OnlineStaff : <Staff>	List of online staff
ActiveStaff : <Staff>	List of online staff currently chatting
Methods	
beginChat(Staff,Account): void	Creates a new account involving a staff and a buyer or seller
endChat(Staff,Account): void	Closes the chat
setChatActive(Account,Boolean):Account	Changes the state of the chat
setSupporter(Staff,Boolean):void	appoint a staff member to a chat

Staff	
Attributes	
staffID: Integer name: String	
Methods	
respondCustomerSupport(): void	Allows the staff to reply to customer request
searchAccounts(): void	Searches for all the database accounts and returns them
modifyAccount(Account modificatedAccount): void	Allows the staff to change all info of an account
blockAccount(): void	Allows for user blocking if the terms have been not followed
warnAccount(): void	Sends warning for an account that does not follow the terms
sendSystemRequest(int commandCode, String username, integer staffID): void	Requests a system commands to be executed by the system
createAccount(Account,String):Boolean	Creates a new account by updating the DB
- editAccountInfo(Account,Account):Account	Fetches account info and updates it in the DB
+ getAccount(Account):Account	

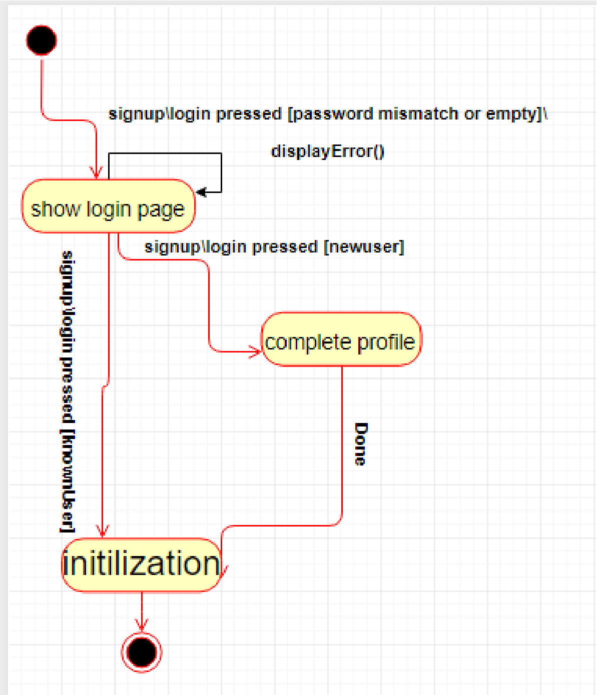
Property	
Attributes	
seller : Account	The account of the hosting seller
buyer : Account	The account of the customer
status : String	The availblity of the property
payment: Due : String	The date of which the payment should be paid before
Address : String	The address of the property
price : float	The price of the property
name : String	The name of the property
type : String	The type of the property
Methods	
+ getInfo(): String	Fetches info from the DB
+ getSeller(): Seller	Rerturns the seller account
+ getCustomer : Account	Return the buyer account

Buyer	
Attributes	
+ NumOfUnitsSold: int	Number of units that have been bought by the buyer
+ NumOfUnitsRented: int	Number of units that have been renting by the customer
+ numberOfWarnings : int	The number of warning that the buyer received
+ isBlocked : boolean	The state of the blockage of the buyer
Methods	
+ subscribe(NotificationList) : void	Allows the buyer to subscribe to a notification list
+ unSubscribe(NotificationList) : void	Al Allows the buyer to unsubscribe to a notification list
+ requsetCustomerSupport() : void	Sends a request for customer support
+ bookAppointment(Date): Boolean	Allows the buyer to book an appointment on a specified date
+ giveFeedBack(Seller,String):void	Allows the buyer to give a feedback on a seller

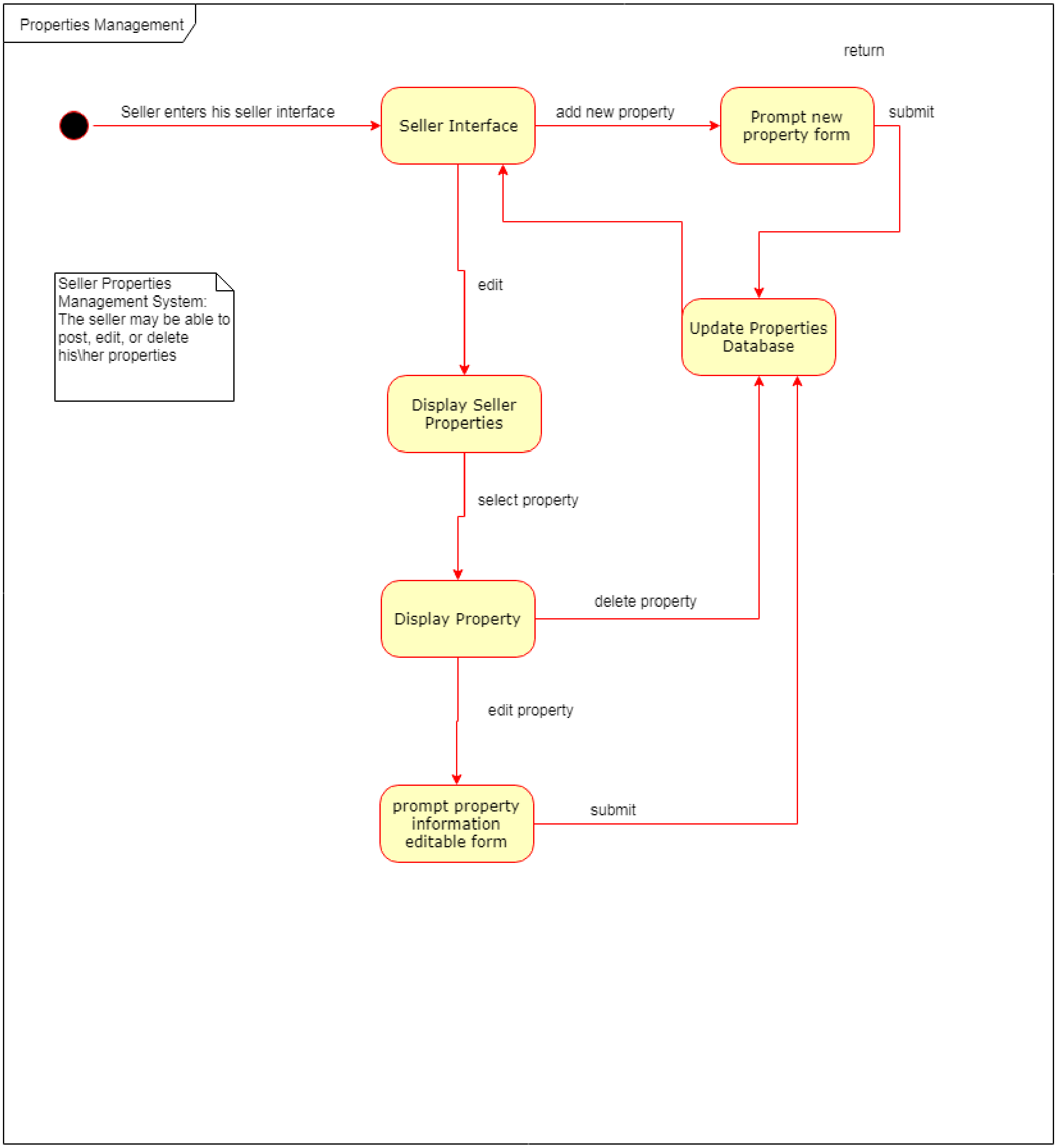
Behavior Design



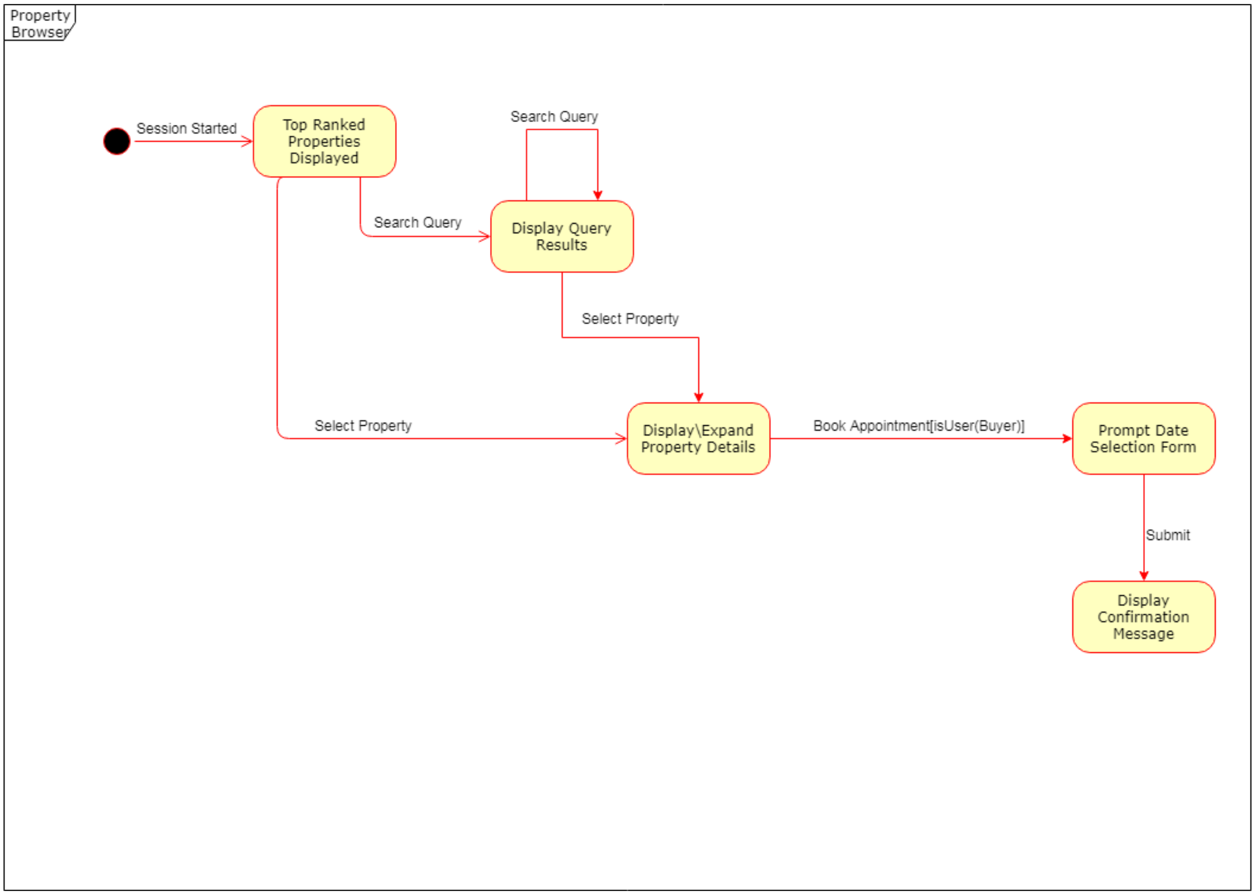
Navigation State Diagram


signup\login pressed [password mismatch or empty]\
displayError()
show login page
signup\login pressed [newuser]
signup\login pressed [knownUser]
complete profile
Done
initilization

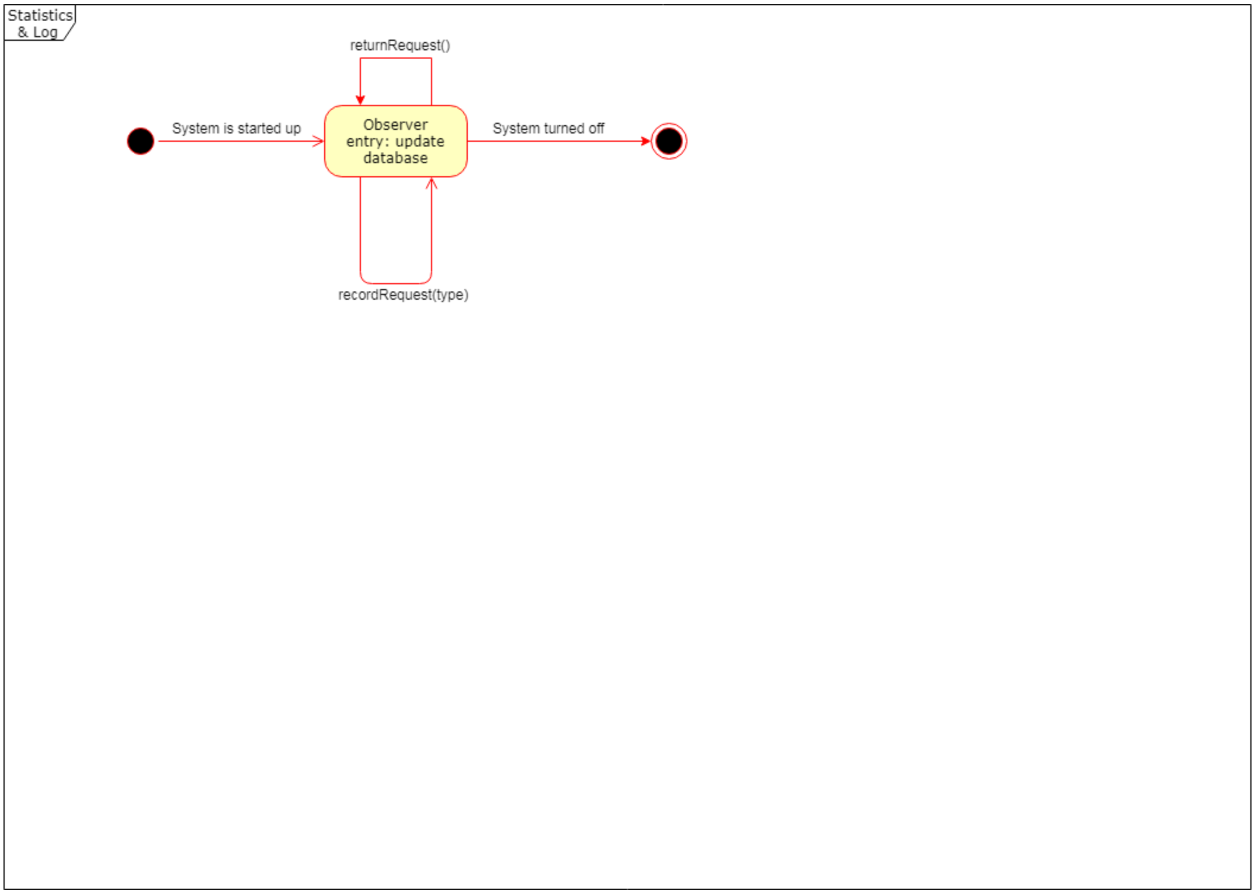
Login State Diagram


Properties Management
return
Seller enters his seller interface
Seller Interface
add new property
Prompt new property form
submit
Seller Properties Management System: The seller may be able to post, edit, or delete his\her properties
edit
Update Properties Database
Display Seller Properties
select property
Display Property
delete property
edit property
prompt property information editable form
submit

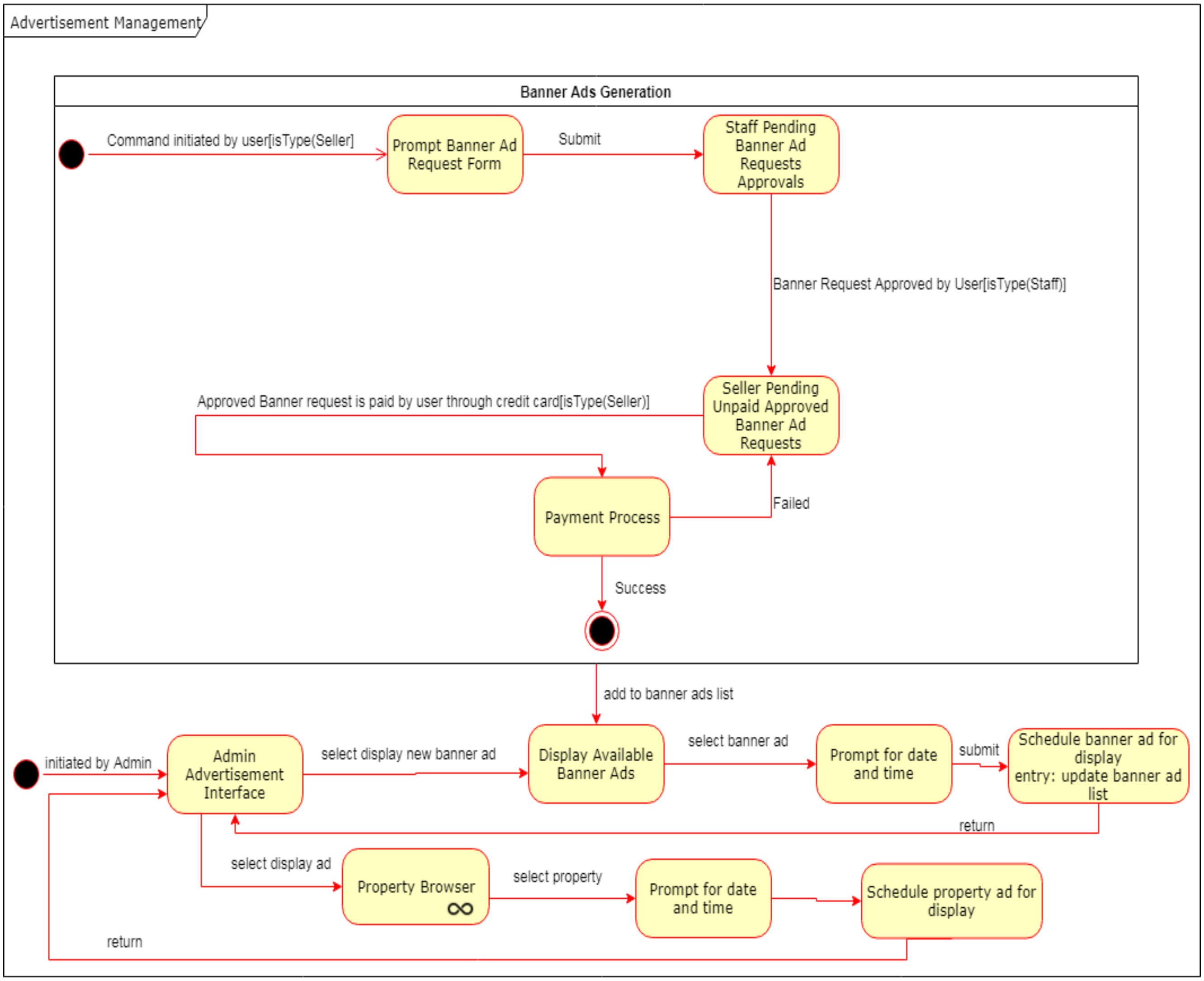
Properties Management State Diagarm


Property Browser
Session Started
Top Ranked Properties Displayed
Search Query
Search Query
Display Query Results
Select Property
Select Property
Display\Expand Property Details
Book Appointment[isUser(Buyer)]
Prompt Date Selection Form
Submit
Display Confirmation Message

Property Browser State Diagram



Statistics & Logs State Diagram


Advertisement Management
Banner Ads Generation
Command initiated by user[isType(Seller)]
Prompt Banner Ad Request Form
Submit
Staff Pending Banner Ad Requests Approvals
Banner Request Approved by User[isType(Staff)]
Approved Banner request is paid by user through credit card[isType(Seller)]
Seller Pending Unpaid Approved Banner Ad Requests
Payment Process
Failed
Success
add to banner ads list
initiated by Admin
Admin Advertisement Interface
select display new banner ad
Display Available Banner Ads
select banner ad
Prompt for date and time
submit
Schedule banner ad for display entry: update banner ad list
return
select display ad
Property Browser
select property
Prompt for date and time
Schedule property ad for display
return

Advertisement Management State Diagram

Conclusion

This document summarizes the design and behavioral description of the United Real Estate System
The work distribution was as follows:

Work	By
Introduction	Asaad AlGhamdi
Architecture Design	Ryan Gadhi
User Interface	Mohammed AlJarbou
Class Diagram	Ryan Gadhi & Asaad AlGhamdi
Class Diagram Descriptions	Ryan Gadhi
Behavior Design	Asaad AlGhamdi & Mohammed AlJarbou
Conclusion	Asaad AlGhamdi