CMPE 202-02 Term Project

Response status 500 will be sent in all APIs in Network error is encountered with the error string as

‘Network Error’

1. **Static✅**

|  |  |
| --- | --- |
| API Function | API to fetch the static data for the application. |
| Database & Tables/Collections | **Database**: Zillow  **Collection: signup** |
| API Method | **GET** |
| API Route | /housing/staticFetch |
| Authorization Header(Jwt Token in the header) | Not required |
| Request |  |
| Response | Response.status:200 Response.data  {  StateNames: “”  Country:: “”  FlooringType:: “”,  HomeType: “”,  Status: “”,  Role: “”,  Parking: “”,  }  OR  Response.status: 403  Response.data  {  “EmailID already in use for the same role”  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Signup✅**

|  |  |
| --- | --- |
| API Function | Common API for User login. |
| Database & Tables/Collections | **Database**: Zillow  **Collection: signup** |
| API Method | **POST** |
| API Route | /housing/signup |
| Authorization Header(Jwt Token in the header) | Not required |
| Request | {  emailID:””,  Password:””,  Role:””,  Name:”” } |
| Response | Response.status:200 Response.data  {  “Success”  }  OR  Response.status: 403  Response.data  {  “EmailID already in use for the same role”  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Login✅**

|  |  |
| --- | --- |
| API Function | Common API for User login. |
| Database & Tables/Collections | **Database**: Zillow  **Collection: signup, user** |
| API Method | **POST** |
| API Route | /housing/login |
| Authorization Header(Jwt Token in the header) | Not required |
| Request | {  EmailID:””  Password:””  Role:”” } |
| Response | Response.status:200 Response.data  {  “JWT token”  }  OR  Response.status: 202  Response.data  {  “Your application status is pending”  }  Response.status: 401  Response.data  {  ‘Invalid Credentials Or Application has been Rejected’  } |

1. **Logout ✅**

|  |  |
| --- | --- |
| API Function | This API is used for user logout for all the three personas |
| Database & Tables/Collections | **-** |
| API Method | **POST** |
| API Route | /logout |
| Authorization Header | Not required |
| Request | - |
| Response | Response.status 200  Response.data  {  ‘Logged out’  } |

1. **Fetch Signup List**

|  |  |
| --- | --- |
| API Function | This API is used to fetch user list to approve/ remove users by the admin |
| Database & Tables/Collections | **Collection: signup** |
| API Method | **GET** |
| API Route | /admin/fetchSignup |
| Authorization Header | Yes |
| Request | Status: “All/ Pending/ Approved” |
| Response | Response.status 200  Response.data  [{  EmailID:  Role:  Status:  }] |

1. **Approve/Remove**

|  |  |
| --- | --- |
| API Function | This API is used for to approve/ remove users by the admin |
| Database & Tables/Collections | **Collection: signup** |
| API Method | **POST** |
| API Route | /admin/auditSignup |
| Authorization Header | Yes |
| Request | EmailID:””  Role:”” Status:”” |
| Response | Response.status 200  Response.data  {  Successful:””  } |

1. **Add new Listing by landlord**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by landlord |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /landlord/addListing |
| Authorization Header | Yes |
| Request | ZIP:’’(required)  StreetAddress:’’, (required)  State:’’ (required)  Country:’’(required)  OwnerID:’’,(required)  OwnerName:’’,(required)  Price:’’,(required)  RealtorID:’’,  RelatorName:’’,  Area:’’,(required)  NoOfBedrooms:’’,  NoOfBathrooms:’’,  Flooring Type:’’,  HomeType:’’,  Parking:’’,  Amenities:’’,  LeaseTerms:’’,  AvailabilityDate;’'  SecurityDeposit:’’,  YearBuilt:’’,  AvailableAs:’’,  OpenHouse:’’, |
| Response | Response.status 200  Response.data  {  ListingObject  } |

1. **Add new Listing by realtor**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by realtor |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /realtor/addListing |
| Authorization Header | Yes |
| Request | ZIP:’’(required)  StreetAddress:’’, (required)  State:’’ (required)  Country:’’(required)  OwnerEmail:’’,(required)  Price:’’,(required)  RealtorID:’’,  RelatorName:’’,  Area:’’,(required)  NoOfBedrooms:’’,  NoOfBathrooms:’’,  Flooring Type:’’,  HomeType:’’,  Parking:’’,  Amenities:’’,  LeaseTerms:’’,  AvailabilityDate;’'  SecurityDeposit:’’,  YearBuilt:’’,  AvailableAs:’’,  OpenHouse:’’, |
| Response | Response.status 200  Response.data  {  ListingObject  } |

1. **Fetch Listing for Owner**

|  |  |
| --- | --- |
| API Function | This API is used to fetch the listing by the landlord |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **GET** |
| API Route | /landlord/getListing |
| Authorization Header | Yes |
| Request | OwenrID:””, |
| Response | Response.status 200  Response.data  [{  Listings: array of listings objects ( for realtor)  Or  Listing object ( for owner)  }] |

1. **Fetch listing for Realtor**

|  |  |
| --- | --- |
| API Function | This API is used to fetch the listing by the realtor |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **GET** |
| API Route | /realtor/getListing |
| Authorization Header | Yes |
| Request | RealtorID:””, |
| Response | Response.status 200  Response.data  [{  Listings: array of listings objects ( for realtor)  Or  Listing object ( for owner)  }] |

1. **Update Listing**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by landlord |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /landlord/updateListing |
| Authorization Header | Yes |
| Request | OwnerID:’’,  Listing object ( whatever params gets updated, please ensure to follow the names as in DB document |
| Response | Response.status 200  Response.data  {  ‘Successful’  } |

1. **Update Listing by realtor**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by realtor |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /realtor/updateListing |
| Authorization Header | Yes |
| Request | RealtorID:’’,  Listing object ( whatever params gets updated, please ensure to follow the names as in DB document along with the listing id as \_id) |
| Response | Response.status 200  Response.data  {  ‘Successful’  } |

1. **Delete Listing by Landlord**

|  |  |
| --- | --- |
| API Function | This API is used to delete a listing by landlord |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **DELETE** |
| API Route | /landlord/deleteListing |
| Authorization Header | Yes |
| Request | OwnerID:’’,  ListingID in the name of \_id |
| Response | Response.status 200  Response.data  {  ‘Successful’  } |

1. **Delete Listing by Realtor**

|  |  |
| --- | --- |
| API Function | This API is used to delete a listing by realtor |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **DELETE** |
| API Route | /realtor/deleteListing |
| Authorization Header | Yes |
| Request | RealtorID:’’,  ListingID in the name of \_id |
| Response | Response.status 200  Response.data  {  ‘Successful’  } |

1. **Get Rent Applications by owner**

|  |  |
| --- | --- |
| API Function | This API is used to get rent applications |
| Database & Tables/Collections | **Collection: application** |
| API Method | **GET** |
| API Route | /landlord/getApplications |
| Authorization Header | Yes |
| Request | OwnerID:  ListingID: Filter: ‘All, Rejected,Approved’ |
| Response | Response.status 200  Response.data  {  Array of applicationObject  } |

1. **Get Rent Applications by realtor**

|  |  |
| --- | --- |
| API Function | This API is used to get rent applications |
| Database & Tables/Collections | **Collection: application** |
| API Method | **GET** |
| API Route | /realtor/getApplications |
| Authorization Header | Yes |
| Request | RealtorID:  ListingID:  Filter: ‘All, Rejected,Approved’ |
| Response | Response.status 200  Response.data  {  Array of applicationObject  } |

1. **Approve/Reject Rent Applications by landlord**

|  |  |
| --- | --- |
| API Function | This API is used to approve/ reject the application by landlord |
| Database & Tables/Collections | **Collection: application** |
| API Method | **POST** |
| API Route | /landlord/getApplications |
| Authorization Header | Yes |
| Request | LandlordID:  Status:  ApplicationID: (will be under the name \_id)  ListingID: |
| Response | Response.status 200  Response.data  {  Array of applicationObject  } |

1. **Approve/Reject Rent Applications by realtor**

|  |  |
| --- | --- |
| API Function | This API is used to approve/ reject the application by realtor |
| Database & Tables/Collections | **Collection: application** |
| API Method | **POST** |
| API Route | /realtor/getApplications |
| Authorization Header | Yes |
| Request | RealtorID:  Status:  ApplicationID: (will be under the name \_id)  ListingID: |
| Response | Response.status 200  Response.data  {  Array of applicationObject  } |

1. **Send applications by renter**

|  |  |
| --- | --- |
| API Function | This API is used to file an application by the renter |
| Database & Tables/Collections | **Collection: application** |
| API Method | **POST** |
| API Route | /renter/fileApplication |
| Authorization Header | Yes |
| Request | ListingID: ‘’,  ApplicantID:’’,  ApplicantName:’’,  ApplicantEmail:’’,  OwnerID:’’,  RealtorID:’’,  PriceQuoted:’’,  Message:’’,  CreditScore:’’,  EmpInformation:’’, |
| Response | Response.status 200 |

1. **Add new Listing by Seller**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by Seller |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /seller/addListing |
| Authorization Header | Yes |
| Request | ZIP:’’(required)  StreetAddress:’’, (required)  State:’’ (required)  Country:’’(required)  OwnerID:’’,(required)  OwnerName:’’,(required)  Price:’’,(required)  RealtorID:’’,  RelatorName:’’,  Area:’’,(required)  NoOfBedrooms:’’,  NoOfBathrooms:’’,  Flooring Type:’’,  HomeType:’’,  Parking:’’,  Amenities:’’,  LeaseTerms:’’,  AvailabilityDate;’'(required)  SecurityDeposit:’’,  YearBuilt:’’,  AvailableAs:’’,  OpenHouse:’’, |
| Response | Response.status 200  Response.data  {  Listings object  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Fetch Listing for Seller / Realtor**

|  |  |
| --- | --- |
| API Function | This API is used to fetch the listing by the seller |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **GET** |
| API Route | /seller/getListing |
| Authorization Header | Yes |
| Request | OwenrID:””, |
| Response | Response.status 200  Response.data  {  Listings: array of listings objects  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Update Listing by Seller**

|  |  |
| --- | --- |
| API Function | This API is used to add new listing by seller |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **POST** |
| API Route | /seller/updateListing |
| Authorization Header | Yes |
| Request | OwnerID:’’,  ListingID:  Listing object ( whatever params gets updated, please ensure to follow the names as in DB document |
| Response | Response.status 200  Response.data  {  ‘Successful’  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Delete Listing by Seller**

|  |  |
| --- | --- |
| API Function | This API is used to delete a listing by Seller |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **DELETE** |
| API Route | /seller/deleteListing |
| Authorization Header | Yes |
| Request | OwnerID:’’,  ListingID in the name of \_\_id |
| Response | Response.status 200  Response.data  {  ‘Successful’  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Get Rent Applications by Seller**

|  |  |
| --- | --- |
| API Function | This API is used to get buy applications |
| Database & Tables/Collections | **Collection: application** |
| API Method | **GET** |
| API Route | /seller/getApplications |
| Authorization Header | Yes |
| Request | OwnerID:  ListingID: Filter: ‘All, Rejected,Approved’ |
| Response | Response.status 200  Response.data  {  Array of applicationObject  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Approve/Reject Applications by Seller**

|  |  |
| --- | --- |
| API Function | This API is used to approve/ reject the application by seller |
| Database & Tables/Collections | **Collection: application** |
| API Method | **POST** |
| API Route | /seller/getApplications |
| Authorization Header | Yes |
| Request | Status:  ApplicationID: (will be under the name \_id)  ListingID: |
| Response | Response.status 200  Response.data  {  Array of applicationObject  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Send applications by buyer**

|  |  |
| --- | --- |
| API Function | This API is used to file an application by the buyer |
| Database & Tables/Collections | **Collection: application** |
| API Method | **POST** |
| API Route | /buyer/fileApplication |
| Authorization Header | Yes |
| Request | ListingID: ‘’,  ApplicantID:’’,  ApplicantName:’’,  ApplicantEmail:’’,  OwnerID:’’,  RealtorID:’’,  PriceQuoted:’’,  Message:’’, |
| Response | Response.status 200  Response.data  {  Array of applicationObject  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Search Listings Search Results Page**

|  |  |
| --- | --- |
| API Function | This API will get the list of listings along with filtered list if mentioned |
| Database & Tables/Collections | **Collection: listing** |
| API Method | **GET** |
| API Route | /user/searchListing |
| Authorization Header | Yes |
| Request | (Ensure to send all the params, if not set for the search send those variables as empty ie “”)  ListingID: ‘’,  Zip:’’,  StreetAddress:’’,  State:’’,  Country:’’,  MinArea:’’,  MaxArea:’’,  MinPrice:’’,  MaxPrice:’’,  NoOfBedrooms:’’,  NoOfBathrooms:’’,  FlooringType:’’,  HomeType:’’,  Parking:’’,  YearBuilt:’’,  AvailableAs:’’, |
| Response | Response.status 200  Response.data  {  Array of Listing Object  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Add a new Favorite Home**

|  |  |
| --- | --- |
| API Function | This API is used to set a home as favorite by user |
| Database & Tables/Collections | **Collection: users** |
| API Method | **POST** |
| API Route | /user/addFavoriteHome |
| Authorization Header | Yes |
| Request | UserID: ‘’,  ListingID: ‘’,  ImageURL:””  StreetAddress:””  ZIP:””  Price:””  NoOfBedrooms:””  NoOfBathrooms:”” |
| Response | Response.status 200  Response.data  {  Favourite Home object  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Get Favorite Homes**

|  |  |
| --- | --- |
| API Function | This API is used to get favorite homes/listings |
| Database & Tables/Collections | **Collection: users** |
| API Method | **GET** |
| API Route | /user/getFavoriteHome |
| Authorization Header | Yes |
| Request | UserID: |
| Response | Response.status 200  Response.data  {  Array of Favorite Home Object  } |

1. **Add a new Favorite Search**

|  |  |
| --- | --- |
| API Function | This API is used to save a search as favorite by user |
| Database & Tables/Collections | **Collection: users** |
| API Method | **POST** |
| API Route | /user/addFavoriteSearch |
| Authorization Header | Yes |
| Request | SearchName:’’,  ZIP:’’,  StreetAddress:’’,  PriceStartRange: ‘’,  PriceEndRange: ‘’,  Area:’’,  NoOfBedrooms:”,  NoOfBathrooms:”,  FlooringType:”,  HomeType:”,  Parking:”,  Amenities:”,  YearBuilt:” |
| Response | Response.status 200  Response.data  {  Favourite Search object  }  OR  Response.status: 500  Response.data  {  “Network Error”  } |

1. **Get Favorite Searches**

|  |  |
| --- | --- |
| API Function | This API is used to get favorite searches |
| Database & Tables/Collections | **Collection: users** |
| API Method | **GET** |
| API Route | /user/getFavoriteSearch |
| Authorization Header | Yes |
| Request | UserID: |
| Response | Response.status 200  Response.data  {  Array of Favorite Search Object  } |