Spring 25 CS5200

Final Project Report

Group 6: Siyuan Liu, Ting Li, Yunyu Guo

April 15 2025

Abstract:

This project is to design and implement a database and application for the team OAKhoury Trees who is working with the non-profit organization Trees for Oakland contributing to the urban forest in Oakland by planting and maintaining trees.

This project designs a data model with 10 classes: user, neighborhood, volunteer, permit, plantingZone, treeSpecies, treeRequest, treePlanting, volunteerPlant, siteVisit; maps it to the relational database model, creates the database in MySQL, populates the database including real data of Oakland neighborhood, zip code, city approved tree species and relative characteristics, write a set of SQL queries, and then implement a user interface to execute these queries using Java.

The solutions are delivered by developing an application which allows residents to submit requests to plant a tree, residents and volunteers to sign up, and an administration team to process the requests, schedule the site visit and recommend a tree by checking tree inventory and planting requirements, and schedule the tree planting event with volunteers.

Problem Description:

***Entities**:*

- User: superclass stores common attributes about all the users (volunteer, admin, resident), including first name, last name, email, password, zip code and role. Resident and admin do not have additional attributes or unique behaviors, so we decided not to create separate subclasses for them. Instead, we use a role attribute in the User class to distinguish between Resident, Volunteer, and Admin. Only the volunteer has extra attributes thus a Volunteer subclass is used.
- Volunteer: subclass of User, includes application status, and availability.
- Permit: A resident needs to upload a tree planting permit on the tree request application. Permit class contains permit ID, permit status, and issue date.
- TreeRequest: manages tree planting requests with details such as street address, request zip code, payment amount, relationship to property, date submitted, request status and phone. Once a treeRequest is created, an admin should visit the site to gather more details.
- SiteVisit: logs site visits for tree planting requests, including visit date, status, whether the planting location is under power line, minimum bed width, and photo of the site before tree planting. The attributes isPowerLine (boolean) and bedWidth correspond to the attributes in the Tree class (isPlantableUnderPowerlines, minPlantingBedWidth). These are used to recommend suitable trees to the planting site by admin.
- TreePlanting: includes plant date, and photo after planting. TreePlanting class also associates with TreeSpecies class which relationship will be used to track tree species and the number of trees planted in different neighborhoods.
- VolunteerPlant: the volunteers will help with the tree planting job assigned by an admin. This class tracks volunteer participation in tree planting events, including work hours and workload feedback.

- PlantingZone: attribute factor indicating whether the tree can be planted in harsh sites(windy, dry, or salty), near bay locations, in highly urbanized zones, or in residential places adjacent to natural areas. A lookup table is used to include all the factors.
- TreeSpecies: contains information about different tree species, including common and scientific names, height, width, and other characteristics, and inventory. The tree species is generated based on the City of Oakland Street Tree List. Admin manages the TreeSpecies, including adding or deleting a tree species and making changes to the inventory of one tree species.
- Neighborhood: stores neighborhood name, description and district.

Assumptions and Justifications for Key Selections:

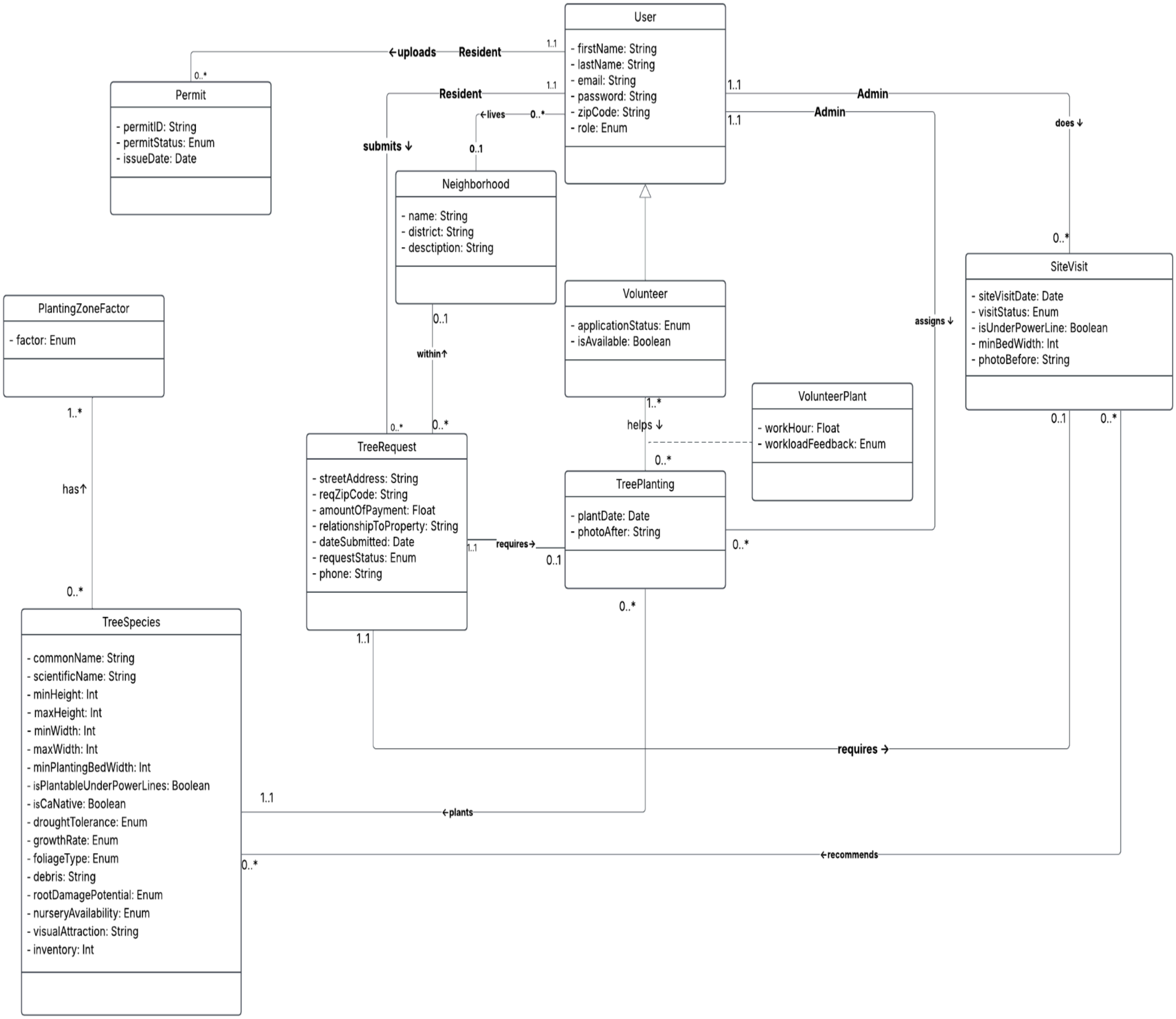
- Related assumption about CK in treeRequests scheme: Though a registered user can submit multiple treeRequest, the same day, the same phone number user with the same street address can only submit one treeRequest.
- Related assumption: Justification for SK in treeRequests: We use a surrogate key (requestID) to keep the table simple and easy to reference. Composite Candidate Key would require using multiple columns in joins, which makes queries more complex. Also, surrogate keys don't change, while fields like "phone" might change, which causes more data modifications.
- Justification for SK in treeSpecies: we use SK treeID instead of two candidate key, commonName and scientificName: there is 64 different trees in total, indicating 2-digits Int which requires around 1 bytes for storage, while normally all the commonName and scientificName are above 10 chars, which requires more than 10 bytes. Besides, some scientificName like "Quercus sartorii" are Latin not English, which might be easily mis-spelt.
- SK plantID in treePlantings scheme: junction table volunteerPlants also requires a unique PK to identify the planting event, volunteer id vid itself cannot be unique identifier if the same volunteers attend many tree planting events, thus surrogate key plantID is copied in the junction table volunteerPlants as a composite PK
- Volunteer can only see the latest status of its own application. Only store the latest availability of volunteers when they sign up for treePlanting event.
- Resident can require a tree to be planted at residential address, and can require a tree to be planted at a business address, so zip code in the user class could be different from the zip code in the TreeRequest class. Resident can submit more than 1 request, a request only links to the specific resident.
- TreeRequest has a phone attribute associated with each request, because the same resident can use a business number when requesting a tree to be planted at a business address, and a personal phone number when requesting a tree to be planted at a residential address.
- A TreeSpecies has at least one PlantingZone, each PlantingZone has at least one corresponding TreeSpecies. Model using a lookup table to store PlantingZone factors names.
- In TreeSpecies class, the following attributes' units are inches in : minHeight, maxHeight, minWidth, maxWidth, minPlantingBedWidth.
- Each TreePlanting event only plants 1 tree, each TreeRequest can only request 1 tree.

- The plantingZoneFactors class includes factors: Harsh sites: windy, dry, or salty; Near bay locations; Highly urbanized zones; Residential adjacent to natural areas.

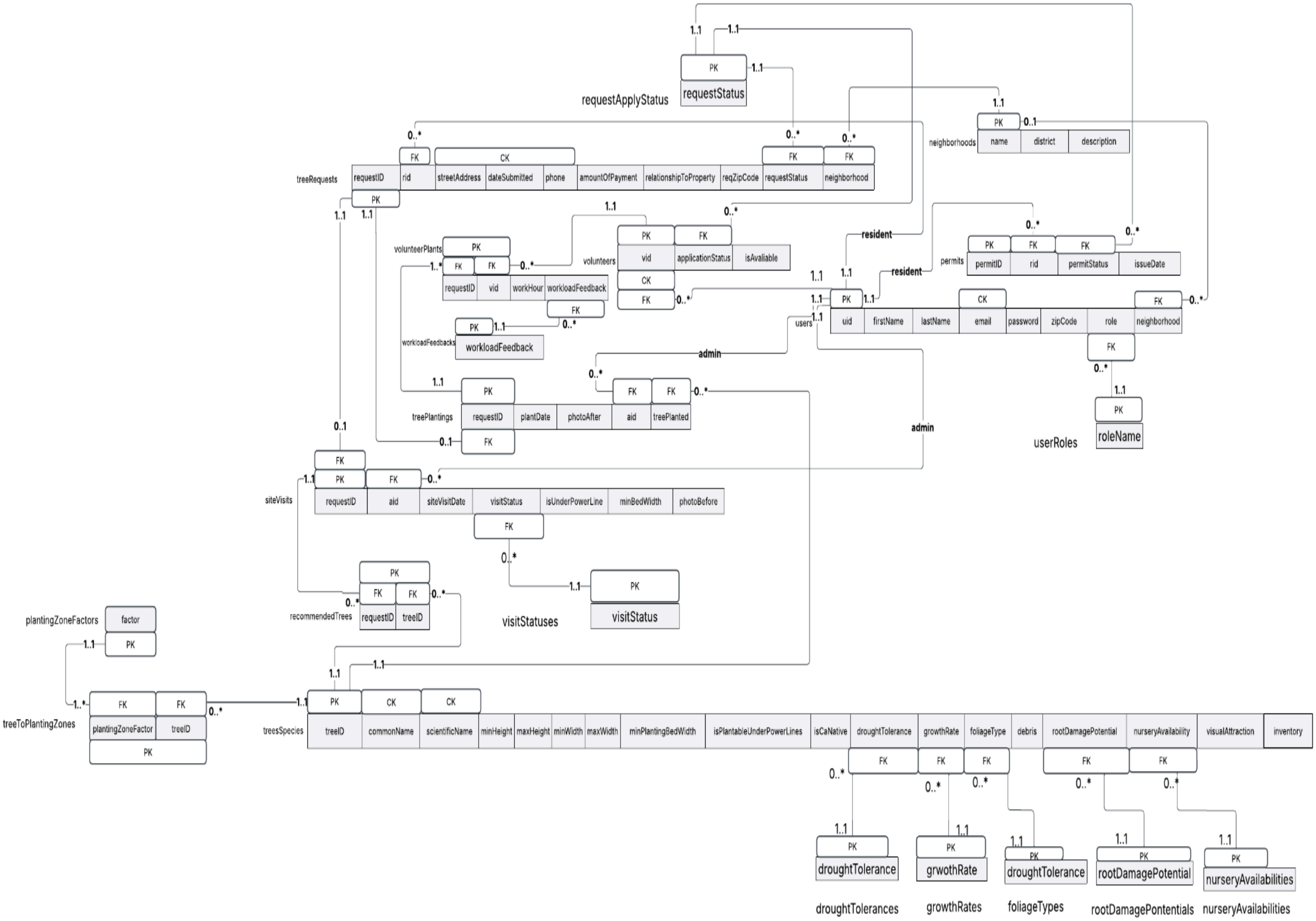
In addition, we reviewed the current tree request form (Google form) and discovered this data collection format would be inefficient for the organization to manage the data. We designed the application to store data and connect the admin team to the tree request by assigning an admin member for site visit before planting a tree. We reviewed the site visit process and criterias of recommending trees to the residents based on factors like clean zone, tree bed width and tree height etc which are used to design the logic of recommending a tree based on the tree planting environment. We reviewed the volunteer sign up page (directed to a third party website MeetUp) and discovered we can incorporate the functions of assigning volunteers to tree planting events, recording volunteer workload and feedback, and storing photos of the tree planted in the application.
We created the neighborhood class which facilitates retrieving neighborhood, tree request and user information in report queries. Because resident and tree request is a many to many relationship, the neighborhood could be different in treeRequests table and users table.

In terms of physical design, both treeRequest class and user class have the same attribute zip code which may seem redundant, but it addresses the situation that resident’s address might be different from submitted treeRequest address.

UML Class Diagram:



RDB Scheme Diagram:



Please see Lucid Chart for UML and scheme

Normalized relations:

Related assumption about CK in treeRequests scheme: Though a registered user can submit multiple treeRequest, the same day, the same phone number user with same street address can only submit one treeRequest.

Justification for SK in treeRequests: We use a surrogate key (requestID) to keep the table simple and easy to reference. Composite Candidate Key would require using multiple columns in joins, which makes queries more complex. Also, surrogate keys don't change, while fields like "phone" might change, which causes more data modifications.

Justification for SK in treeSpecies: we use SK treeID instead of two candidate key: commonName and scientificName, because there are 64 different trees in total, indicating 2-digits Int which requires around 1 bytes for storage, while normally all the commonName and scientificName are above 10 chars, which requires more than 10 bytes. Besides, some scientificName like "Quercus sartorii" are Latin, which might be easily mis-spelt.

Justification for SK in users: we have multiple schemes that need the PK of users as a FK, including volunteers, treeRequests, siteVisits, etc. Although we have email as a CK, keeping the email information in multiple schemes will result in possible privacy risks and this requires a larger space for storage than using just int datatype for the id. Introducing a user id as SK can solve these problems.

Justification for not introducing a SK in scheme neighborhood: since we checked the local wiki about the Oakland neighborhood, we can see that there's no two identical neighborhood names in Oakland, and the neighborhood names are quite short (less than 40 characters). The neighborhood name will not change very frequently. Also, since we have treeRequests and users that need neighborhood information, and we will frequently query the treeRequests and generate reports about the trees planted in neighborhoods, if we introduce a SK will result in frequent joins just to get the neighborhood name. We think this is a time-consuming thing that can be improved by denormalization, which is not introducing SK and just using neighborhood names as the PK.

SK plantID in treePlantings scheme: junction table volunteerPlants also requires a unique PK to identify the planting event, volunteer id vid itself cannot be unique identifier if the same volunteers attend many tree planting events, thus surrogate key plantID is copied in the junction table volunteerPlants as a composite PK.

3NF check:
Schemes
users:
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute
2NF: Every non-prime attribute is fully dependent on the PK and CK
FD in this scheme:

{uid} -> {firstName, lastName, password, zipCode, role, neighborhood}
{email} -> {firstName, lastName, password, zipCode, role, neighborhood}
3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)
There's no transitive dependencies here
volunteers:
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute
2NF: Every non-prime attribute is fully dependent on the PK
FD in this scheme:
{vid} -> {applicationstatus, isAvailable}
3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)
There's no transitive dependencies here
permits:
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute
2NF: Every non-prime attribute is fully dependent on the PK
FD in this scheme:
{permitID} -> {rid, permitStatus, issueDate}
3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)
There's no transitive dependencies here
neighborhoods:
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute
2NF: Every non-prime attribute is fully dependent on the PK
FD in this scheme:
{name} -> {district, description}
3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)
There's no transitive dependencies here
treeRequests:
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute
2NF: Every non-prime attribute is fully dependent on the PK and CK
FD in this scheme:
requestID -> {rid, amountOfPayment, relationshipToProperty, reqZipCode, requestStatus, neighborhood}
3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)
There's no transitive dependencies here

Although it might seem that one street address will determine one zipCode...there's an exception. Places like campus or airport can have multiple zipCode while they have the same street address. So that's not a transitive dependency.

treePlantings:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF: Every non-prime attribute is fully dependent on the PK

FD in this scheme:

{requestID} -> {plantDate, photoAfter, aid, treePlanted}

3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)

There's no transitive dependencies here

volunteerPlants:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF: Every non-prime attribute is fully dependent on the PK

FD in this scheme:

{requestID, vid} -> {workHour, workloadFeedback}

3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)

There's no transitive dependencies here

siteVisits:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF: Every non-prime attribute is fully dependent on the PK

FD in this scheme:

{requestID} -> {aid, siteVisitDate, visitStatus, isUnderPowerLine, minBedWidth, photoBefore}

3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)

There's no transitive dependencies here

recommendedTrees:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF: Every non-prime attribute is fully dependent on the PK

FD in this scheme:

All the attribute in this table is part of the PK, so there's no non-prime attribute

3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)

There's no transitive dependencies here

treesSpecies:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF: Every non-prime attribute is fully dependent on the PK and CK

FD in this scheme:

{treeID} -> {minHeight,maxHeight, minWidth, maxWidth, minPlantingBedWidth, isPlantableUnderPowerLines, isCaNative, droughtTolerance, growthRate, foliageType, debris, rootDamagePotential, nurseryAvailability, visualAttraction, inventory}

3NF: Every non-prime attribute is non-transitively dependent on every key(no transitive dependencies)

There's no transitive dependencies here

plantingZoneFactors:

1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF and 3NF: the only attribute is the PK, so there's no non-prime attribute and satisfies both 2NF and 3NF.

treeToPlantingZones:

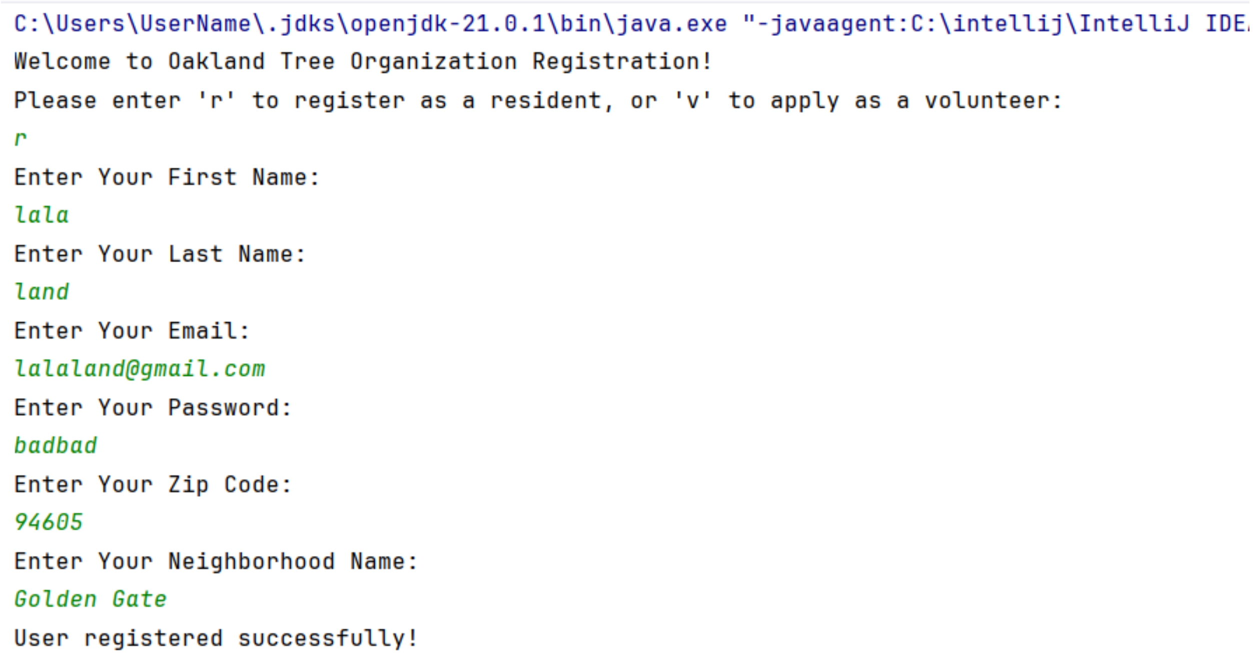
1NF: No composite attribute, all the value of any attribute in a tuple is a single values from the domain of the attribute

2NF and 3NF: all attributes are part of the PK, so there's no non-prime attribute and satisfies both 2NF and 3NF.

Screenshots of running system:

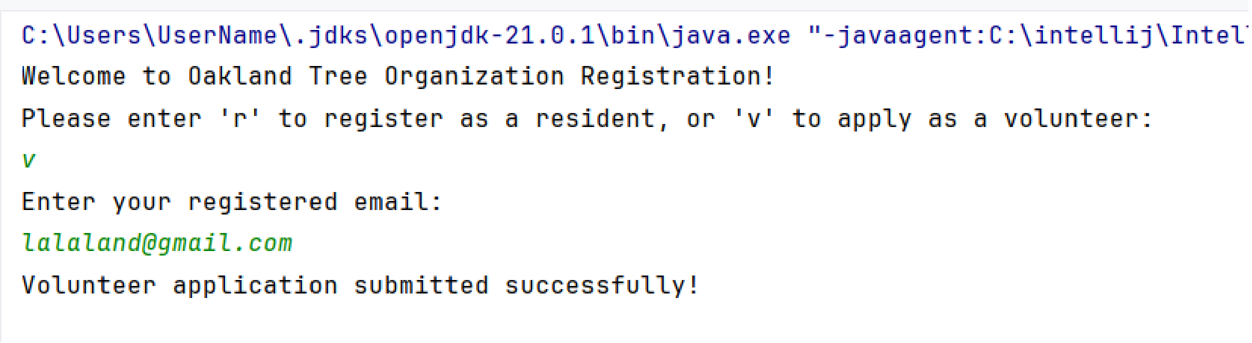
Demonstrating example input/output of each task

Task A1: register a resident

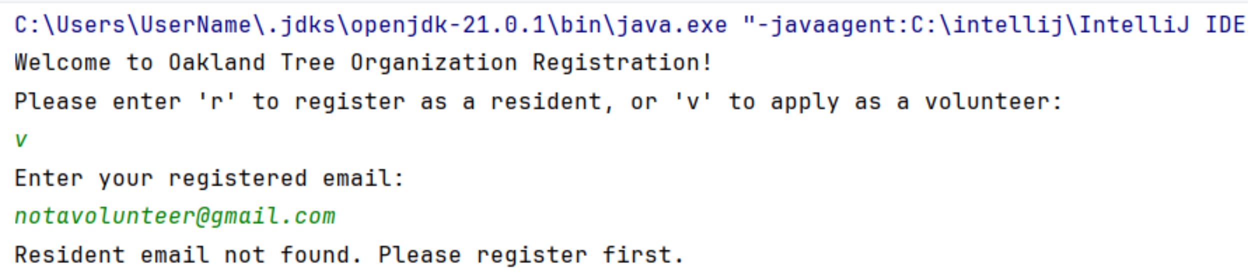


```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDE
Welcome to Oakland Tree Organization Registration!
Please enter 'r' to register as a resident, or 'v' to apply as a volunteer:
r
Enter Your First Name:
lala
Enter Your Last Name:
land
Enter Your Email:
lalaland@gmail.com
Enter Your Password:
badbad
Enter Your Zip Code:
94605
Enter Your Neighborhood Name:
Golden Gate
User registered successfully!
```

Task A2: resident (already registered user) apply to be the volunteer

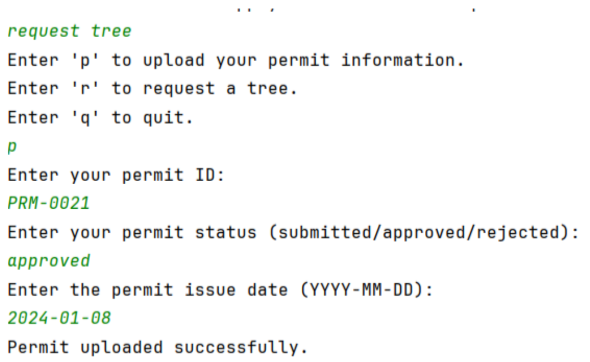


```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\Intel
Welcome to Oakland Tree Organization Registration!
Please enter 'r' to register as a resident, or 'v' to apply as a volunteer:
v
Enter your registered email:
lalaland@gmail.com
Volunteer application submitted successfully!
```



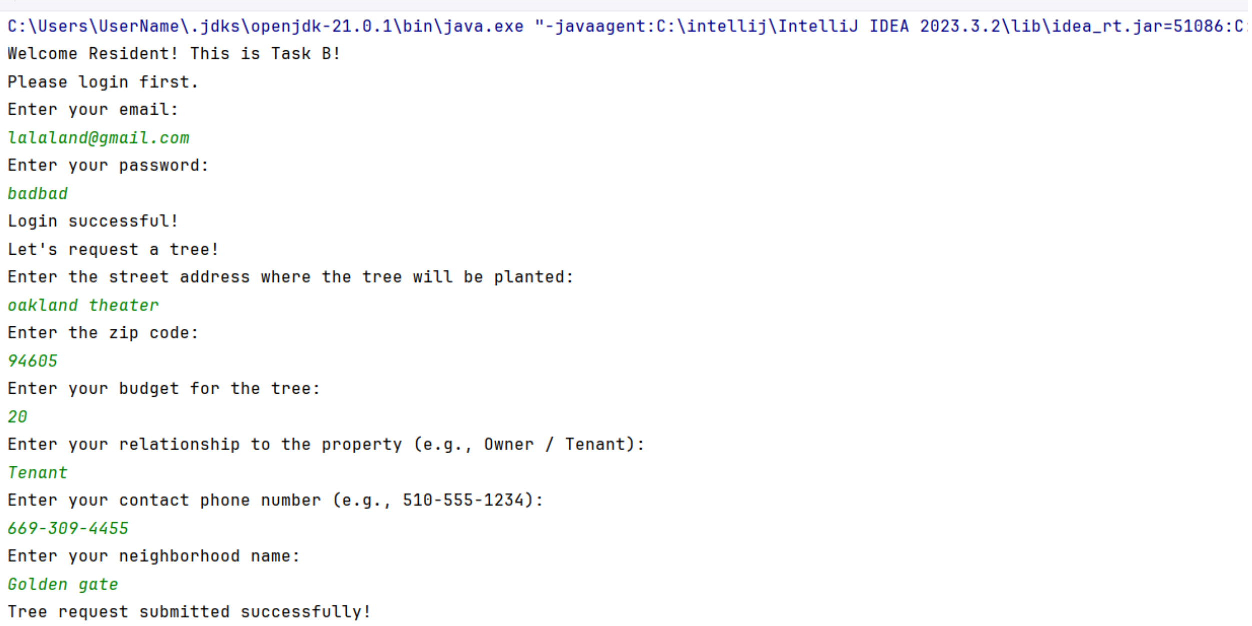
```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDE
Welcome to Oakland Tree Organization Registration!
Please enter 'r' to register as a resident, or 'v' to apply as a volunteer:
v
Enter your registered email:
notavolunteer@gmail.com
Resident email not found. Please register first.
```

Task b1. upload permit before tree request



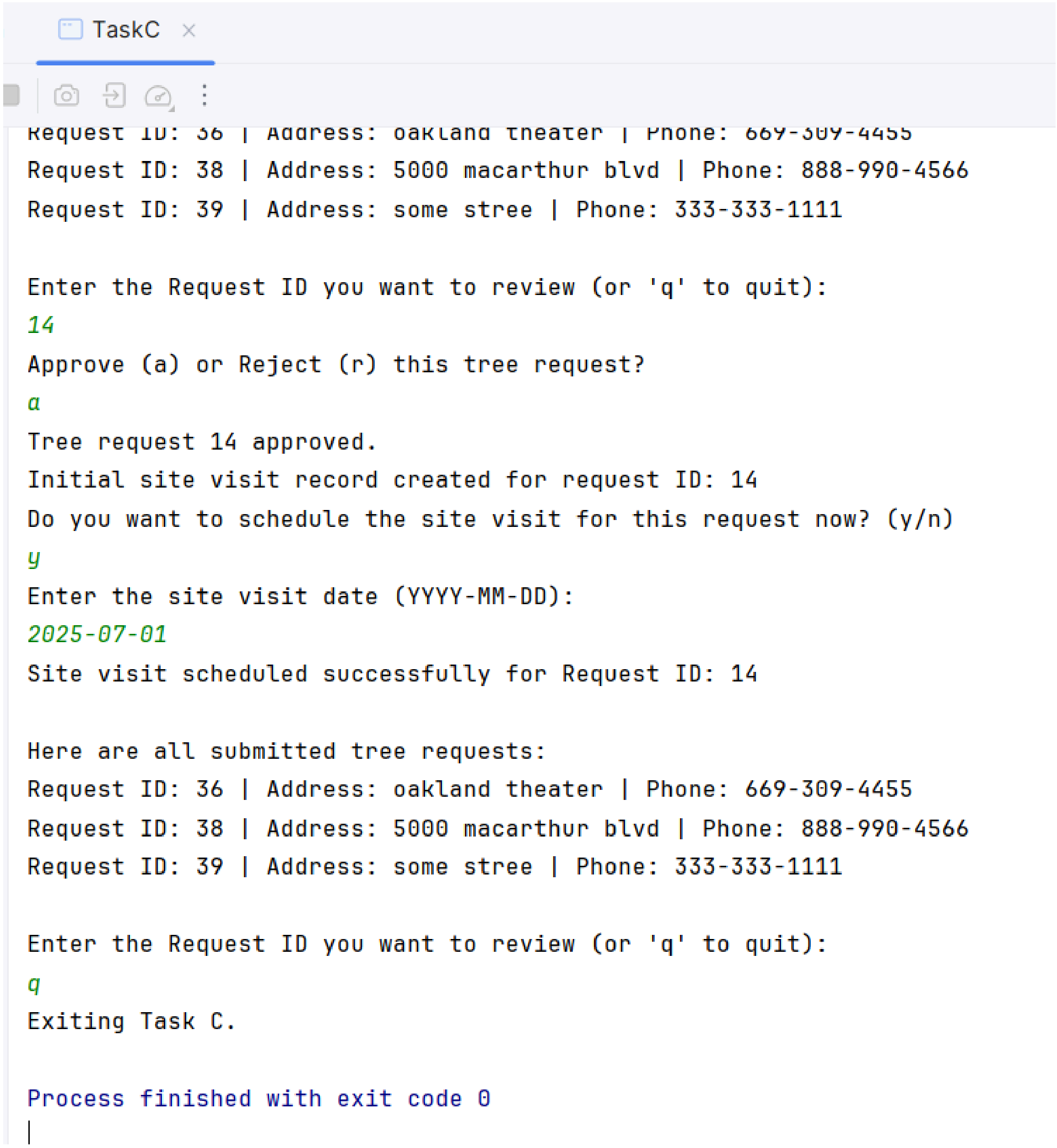
```
request tree
Enter 'p' to upload your permit information.
Enter 'r' to request a tree.
Enter 'q' to quit.
p
Enter your permit ID:
PRM-0021
Enter your permit status (submitted/approved/rejected):
approved
Enter the permit issue date (YYYY-MM-DD):
2024-01-08
Permit uploaded successfully.
```

Task b2. make tree request a tree if has a valid permit



```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_rt.jar=51086:C
Welcome Resident! This is Task B!
Please login first.
Enter your email:
lalaland@gmail.com
Enter your password:
badbad
Login successful!
Let's request a tree!
Enter the street address where the tree will be planted:
oakland theater
Enter the zip code:
94605
Enter your budget for the tree:
20
Enter your relationship to the property (e.g., Owner / Tenant):
Tenant
Enter your contact phone number (e.g., 510-555-1234):
669-309-4455
Enter your neighborhood name:
Golden gate
Tree request submitted successfully!
```

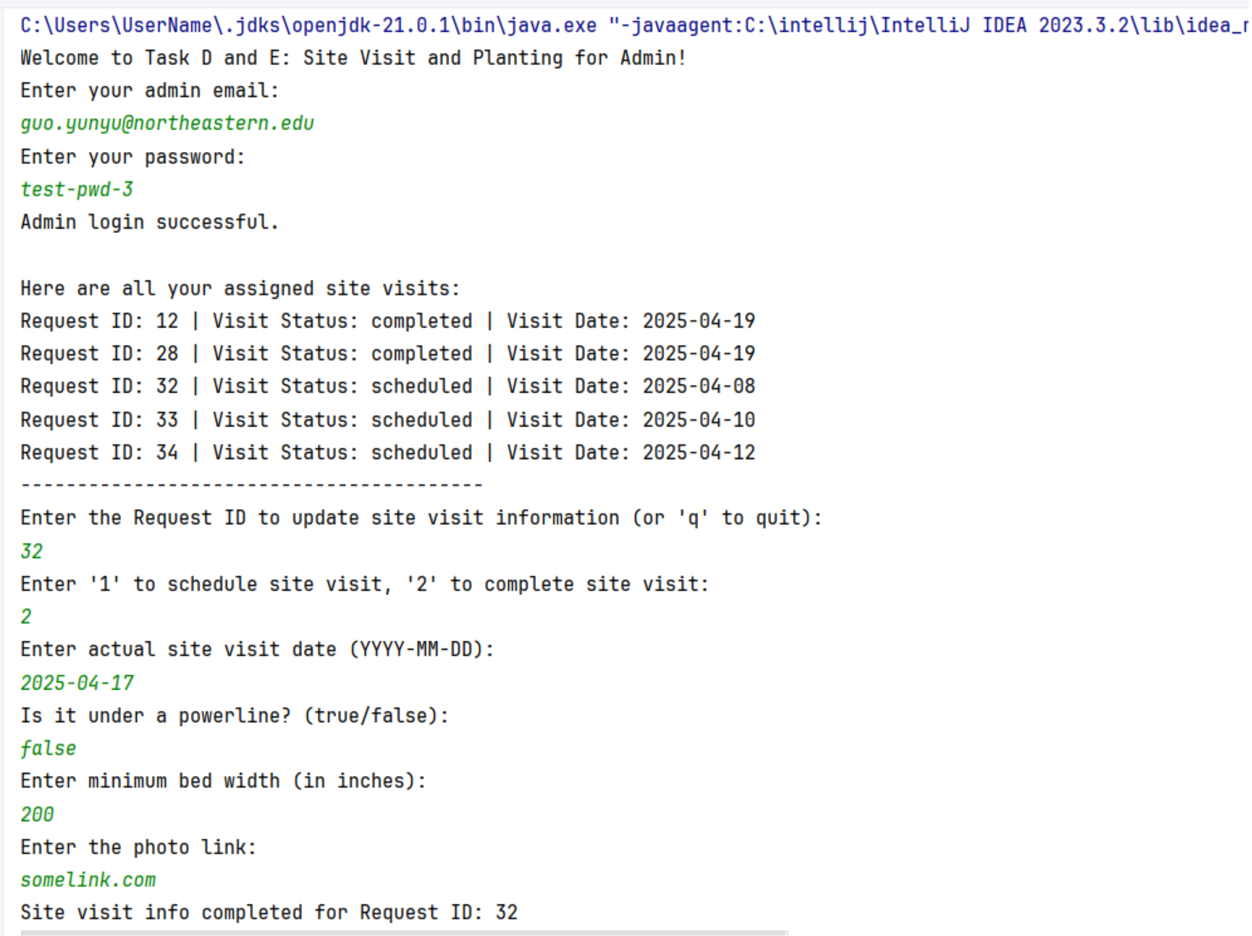
Task c. schedule a visit to a tree



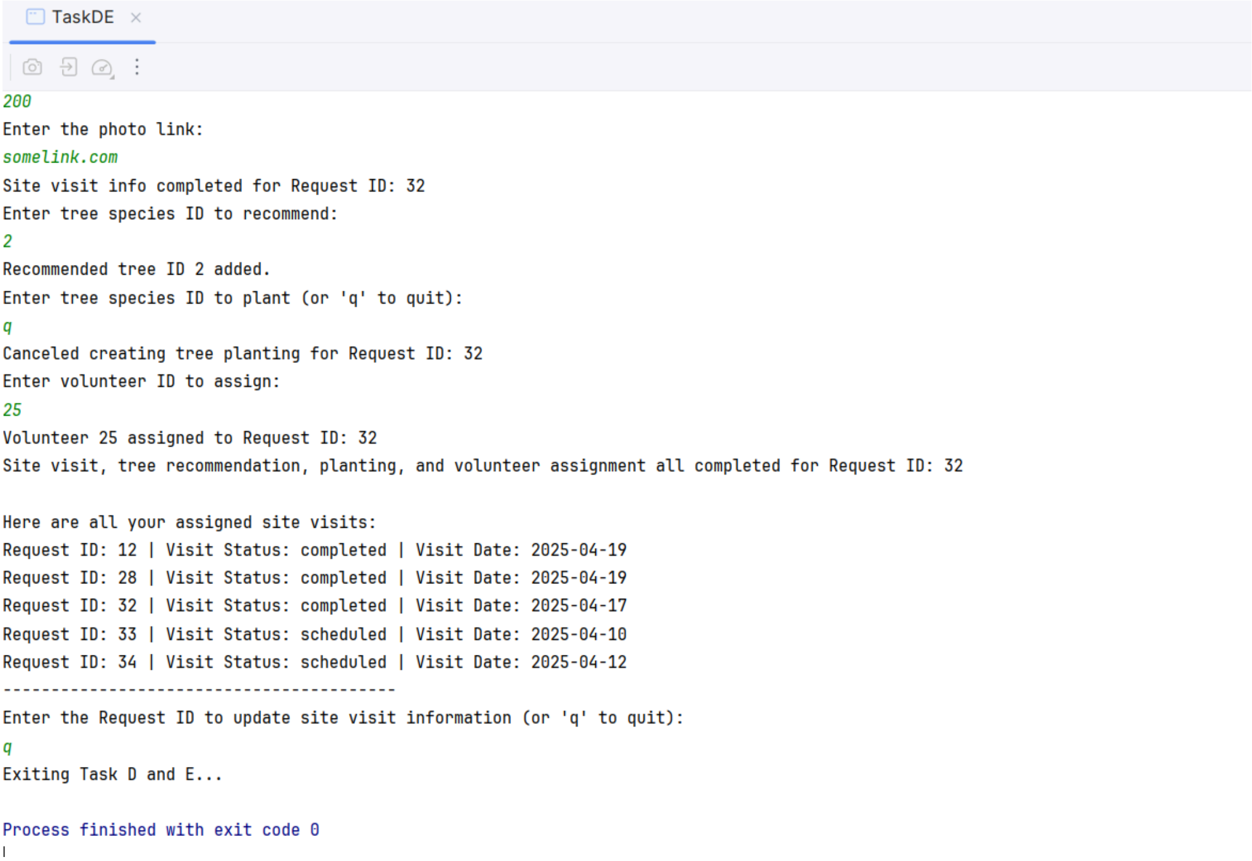
```
Request ID: 36 | Address: oakland theater | Phone: 669-309-4455
Request ID: 38 | Address: 5000 macarthur blvd | Phone: 888-990-4566
Request ID: 39 | Address: some stree | Phone: 333-333-1111

Enter the Request ID you want to review (or 'q' to quit):
14
Approve (a) or Reject (r) this tree request?
a
Tree request 14 approved.
Initial site visit record created for request ID: 14
Do you want to schedule the site visit for this request now? (y/n)
y
Enter the site visit date (YYYY-MM-DD):
2025-07-01
Site visit scheduled successfully for Request ID: 14

Here are all submitted tree requests:
Request ID: 36 | Address: oakland theater | Phone: 669-309-4455
Request ID: 38 | Address: 5000 macarthur blvd | Phone: 888-990-4566
Request ID: 39 | Address: some stree | Phone: 333-333-1111

Enter the Request ID you want to review (or 'q' to quit):
q
Exiting Task C.

Process finished with exit code 0
```

Task D & E On or after a site visit, (admin) record the information gathered by the team including references to photos for the (volunteer) team to review later; (admin) Schedule the planting of a tree, includes booking the team of volunteers who will do the planting:

PS: we combined tasks D and E in the same file since we assumed that when the admin finishes the site visit and tries to update the relevant information, this admin must assign a volunteer after this updating.

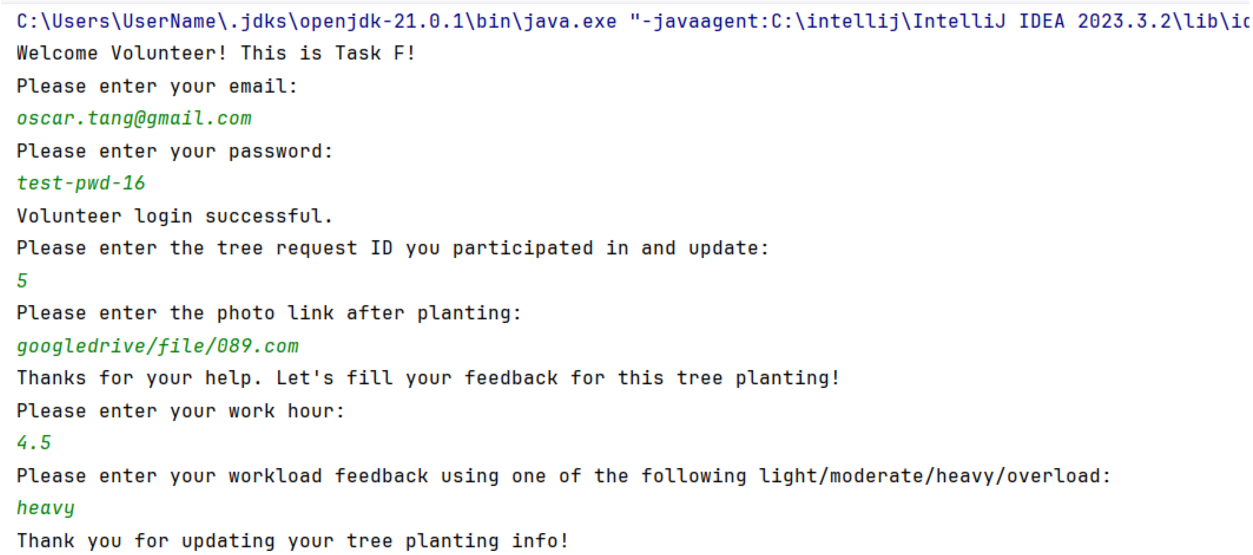


```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_
Welcome to Task D and E: Site Visit and Planting for Admin!
Enter your admin email:
guo.yunyu@northeastern.edu
Enter your password:
test-pwd-3
Admin login successful.

Here are all your assigned site visits:
Request ID: 12 | Visit Status: completed | Visit Date: 2025-04-19
Request ID: 28 | Visit Status: completed | Visit Date: 2025-04-19
Request ID: 32 | Visit Status: scheduled | Visit Date: 2025-04-08
Request ID: 33 | Visit Status: scheduled | Visit Date: 2025-04-10
Request ID: 34 | Visit Status: scheduled | Visit Date: 2025-04-12
-----------------------------------------
Enter the Request ID to update site visit information (or 'q' to quit):
32
Enter '1' to schedule site visit, '2' to complete site visit:
2
Enter actual site visit date (YYYY-MM-DD):
2025-04-17
Is it under a powerline? (true/false):
false
Enter minimum bed width (in inches):
200
Enter the photo link:
somelink.com
Site visit info completed for Request ID: 32
```



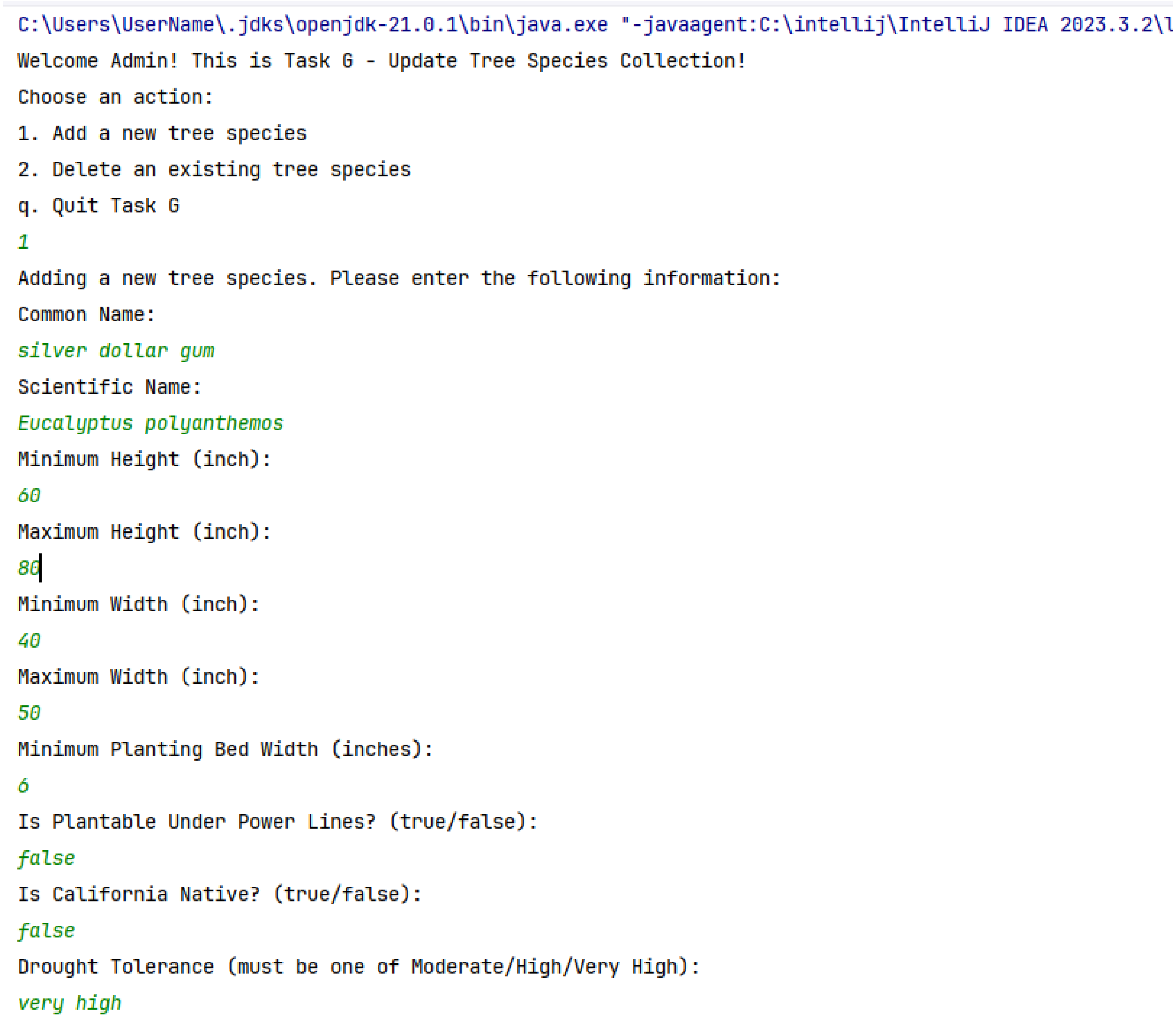
```
200
Enter the photo link:
somelink.com
Site visit info completed for Request ID: 32
Enter tree species ID to recommend:
2
Recommended tree ID 2 added.
Enter tree species ID to plant (or 'q' to quit):
q
Canceled creating tree planting for Request ID: 32
Enter volunteer ID to assign:
25
Volunteer 25 assigned to Request ID: 32
Site visit, tree recommendation, planting, and volunteer assignment all completed for Request ID: 32

Here are all your assigned site visits:
Request ID: 12 | Visit Status: completed | Visit Date: 2025-04-19
Request ID: 28 | Visit Status: completed | Visit Date: 2025-04-19
Request ID: 32 | Visit Status: completed | Visit Date: 2025-04-17
Request ID: 33 | Visit Status: scheduled | Visit Date: 2025-04-10
Request ID: 34 | Visit Status: scheduled | Visit Date: 2025-04-12
----------------------------------------
Enter the Request ID to update site visit information (or 'q' to quit):
q
Exiting Task D and E...

Process finished with exit code 0
```

Task f. After a tree is planted, record volunteers who participated, observations of the tree planting, and references to before/after photos of the site.

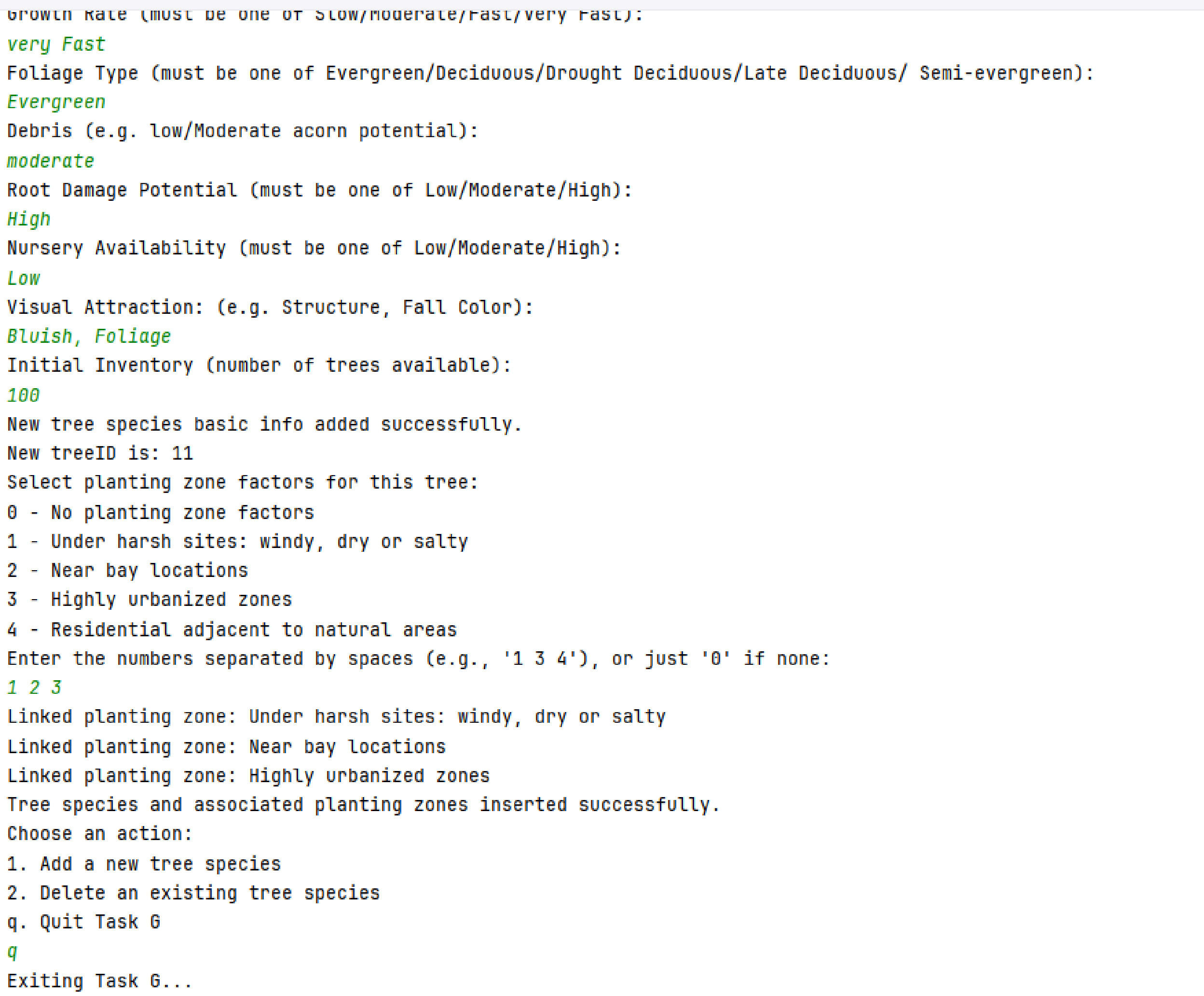


```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\ic
Welcome Volunteer! This is Task F!
Please enter your email:
oscar.tang@gmail.com
Please enter your password:
test-pwd-16
Volunteer login successful.
Please enter the tree request ID you participated in and update:
5
Please enter the photo link after planting:
googledrive/file/089.com
Thanks for your help. Let's fill your feedback for this tree planting!
Please enter your work hour:
4.5
Please enter your workload feedback using one of the following light/moderate/heavy/overload:
heavy
Thank you for updating your tree planting info!
```

Task G. Update the collection of tree species available for planting: removal or addition of species.
G1: add a tree species

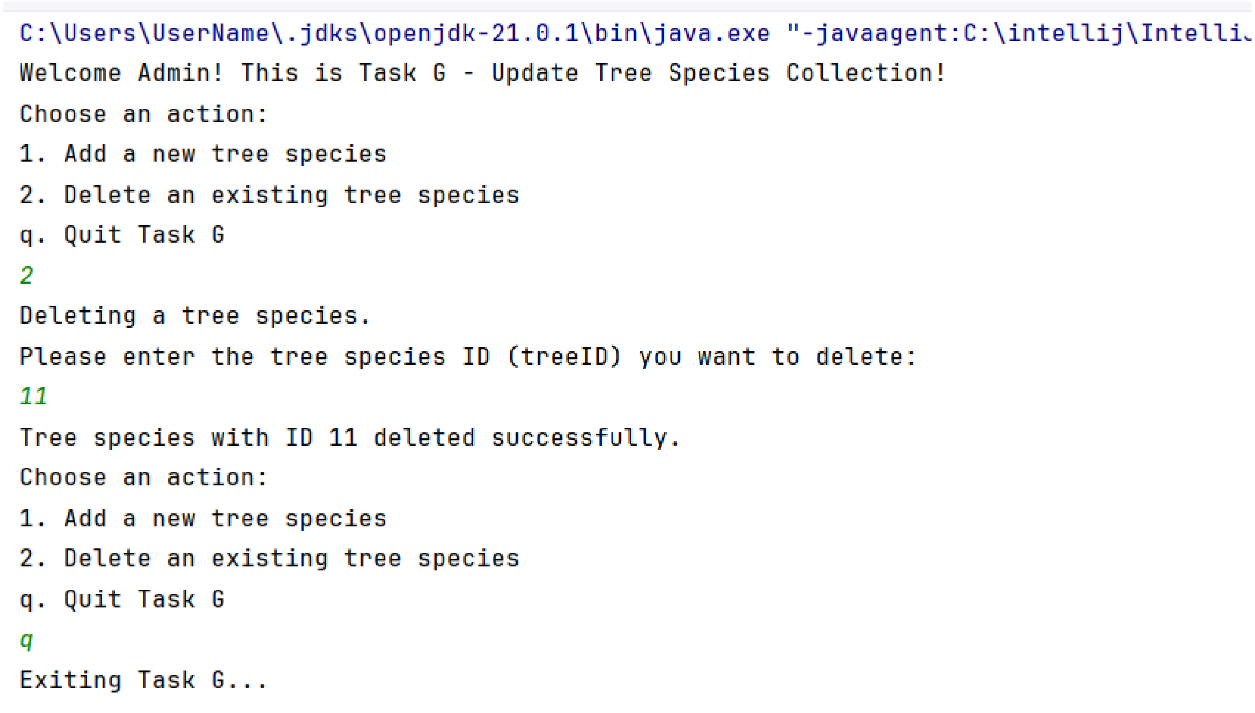


```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\l
Welcome Admin! This is Task G - Update Tree Species Collection!
Choose an action:
1. Add a new tree species
2. Delete an existing tree species
q. Quit Task G
1
Adding a new tree species. Please enter the following information:
Common Name:
silver dollar gum
Scientific Name:
Eucalyptus polyanthemos
Minimum Height (inch):
60
Maximum Height (inch):
80
Minimum Width (inch):
40
Maximum Width (inch):
50
Minimum Planting Bed Width (inches):
6
Is Plantable Under Power Lines? (true/false):
false
Is California Native? (true/false):
false
Drought Tolerance (must be one of Moderate/High/Very High):
very high
```



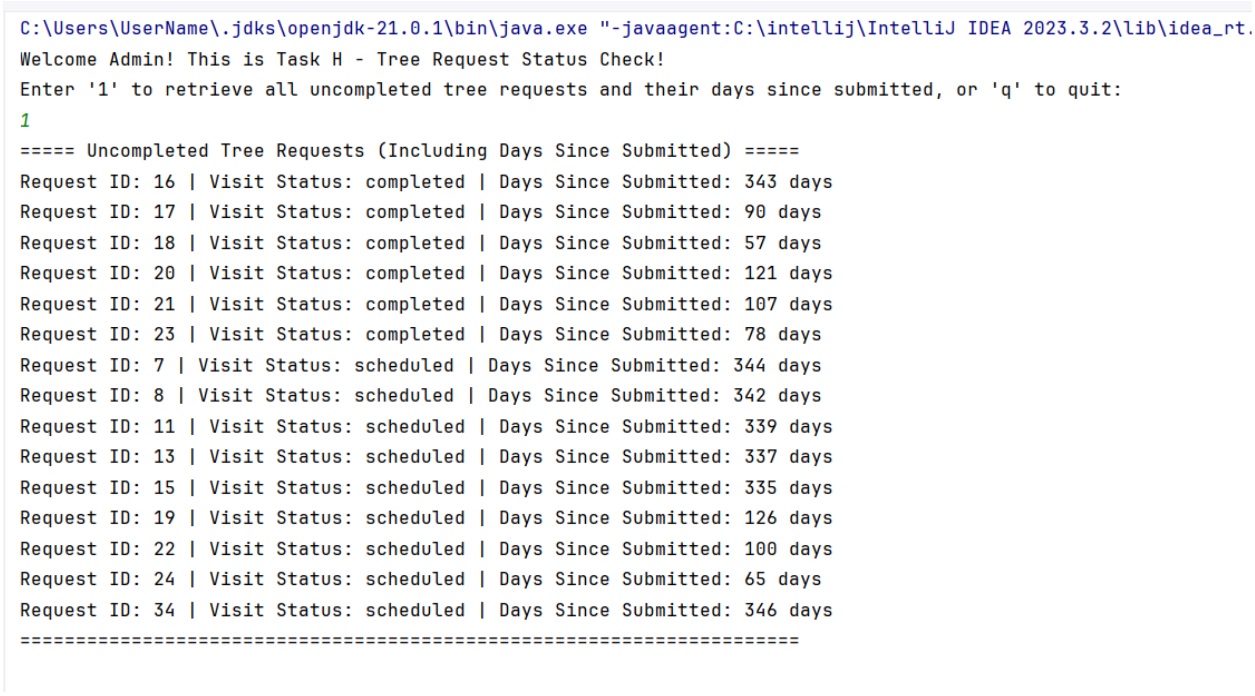
```
Growth Rate (must be one of Slow/Moderate/Fast/Very Fast):
very Fast
Foliage Type (must be one of Evergreen/Deciduous/Drought Deciduous/Late Deciduous/ Semi-evergreen):
Evergreen
Debris (e.g. low/Moderate acorn potential):
moderate
Root Damage Potential (must be one of Low/Moderate/High):
High
Nursery Availability (must be one of Low/Moderate/High):
Low
Visual Attraction: (e.g. Structure, Fall Color):
Bluish, Foliage
Initial Inventory (number of trees available):
100
New tree species basic info added successfully.
New treeID is: 11
Select planting zone factors for this tree:
0 - No planting zone factors
1 - Under harsh sites: windy, dry or salty
2 - Near bay locations
3 - Highly urbanized zones
4 - Residential adjacent to natural areas
Enter the numbers separated by spaces (e.g., '1 3 4'), or just '0' if none:
1 2 3
Linked planting zone: Under harsh sites: windy, dry or salty
Linked planting zone: Near bay locations
Linked planting zone: Highly urbanized zones
Tree species and associated planting zones inserted successfully.
Choose an action:
1. Add a new tree species
2. Delete an existing tree species
q. Quit Task G
q
Exiting Task G...
```

G2. Delete a tree species



```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\Intelli
Welcome Admin! This is Task G - Update Tree Species Collection!
Choose an action:
1. Add a new tree species
2. Delete an existing tree species
q. Quit Task G
2
Deleting a tree species.
Please enter the tree species ID (treeID) you want to delete:
11
Tree species with ID 11 deleted successfully.
Choose an action:
1. Add a new tree species
2. Delete an existing tree species
q. Quit Task G
q
Exiting Task G...
```

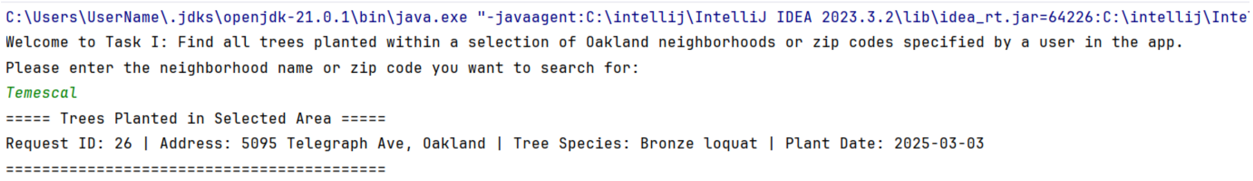
Task H. For all requests to plant a tree that have not yet been completed, show its status, and the number of days that have transpired since it was first submitted.



```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_rt.
Welcome Admin! This is Task H - Tree Request Status Check!
Enter '1' to retrieve all uncompleted tree requests and their days since submitted, or 'q' to quit:
1
===== Uncompleted Tree Requests (Including Days Since Submitted) =====
Request ID: 16 | Visit Status: completed | Days Since Submitted: 343 days
Request ID: 17 | Visit Status: completed | Days Since Submitted: 90 days
Request ID: 18 | Visit Status: completed | Days Since Submitted: 57 days
Request ID: 20 | Visit Status: completed | Days Since Submitted: 121 days
Request ID: 21 | Visit Status: completed | Days Since Submitted: 107 days
Request ID: 23 | Visit Status: completed | Days Since Submitted: 78 days
Request ID: 7 | Visit Status: scheduled | Days Since Submitted: 344 days
Request ID: 8 | Visit Status: scheduled | Days Since Submitted: 342 days
Request ID: 11 | Visit Status: scheduled | Days Since Submitted: 339 days
Request ID: 13 | Visit Status: scheduled | Days Since Submitted: 337 days
Request ID: 15 | Visit Status: scheduled | Days Since Submitted: 335 days
Request ID: 19 | Visit Status: scheduled | Days Since Submitted: 126 days
Request ID: 22 | Visit Status: scheduled | Days Since Submitted: 100 days
Request ID: 24 | Visit Status: scheduled | Days Since Submitted: 65 days
Request ID: 34 | Visit Status: scheduled | Days Since Submitted: 346 days
======================================================================
```

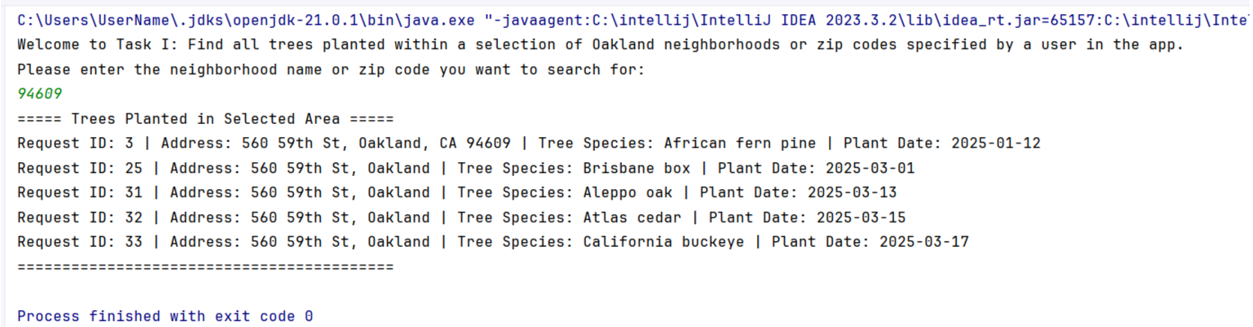
Task I: Find all trees planted within a selection of Oakland neighborhoods or zip codes specified by a user in the app.

I1:



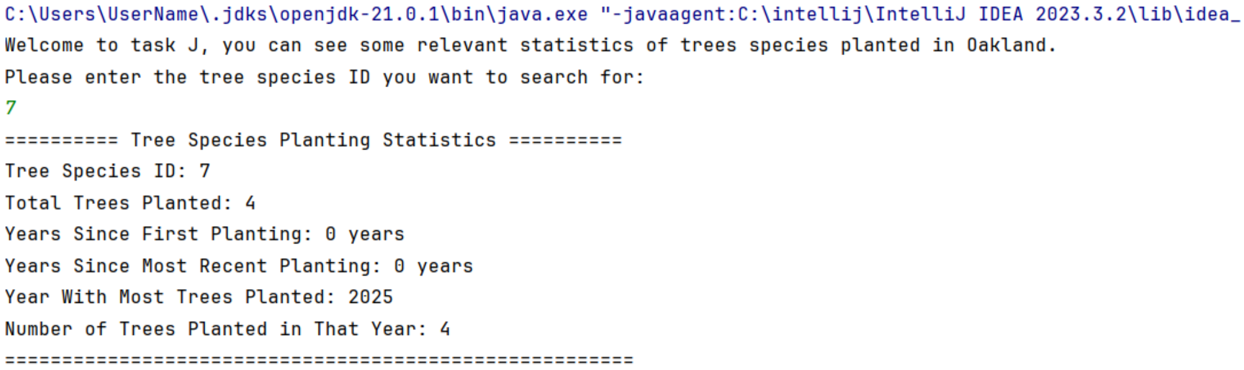
```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_rt.jar=64226:C:\intellij\Inte
Welcome to Task I: Find all trees planted within a selection of Oakland neighborhoods or zip codes specified by a user in the app.
Please enter the neighborhood name or zip code you want to search for:
Temescal
===== Trees Planted in Selected Area =====
Request ID: 26 | Address: 5095 Telegraph Ave, Oakland | Tree Species: Bronze loquat | Plant Date: 2025-03-03
==========================================
```

I2:



```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_rt.jar=65157:C:\intellij\Inte
Welcome to Task I: Find all trees planted within a selection of Oakland neighborhoods or zip codes specified by a user in the app.
Please enter the neighborhood name or zip code you want to search for:
94609
===== Trees Planted in Selected Area =====
Request ID: 3 | Address: 560 59th St, Oakland, CA 94609 | Tree Species: African fern pine | Plant Date: 2025-01-12
Request ID: 25 | Address: 560 59th St, Oakland | Tree Species: Brisbane box | Plant Date: 2025-03-01
Request ID: 31 | Address: 560 59th St, Oakland | Tree Species: Aleppo oak | Plant Date: 2025-03-13
Request ID: 32 | Address: 560 59th St, Oakland | Tree Species: Atlas cedar | Plant Date: 2025-03-15
Request ID: 33 | Address: 560 59th St, Oakland | Tree Species: California buckeye | Plant Date: 2025-03-17
==========================================

Process finished with exit code 0
```

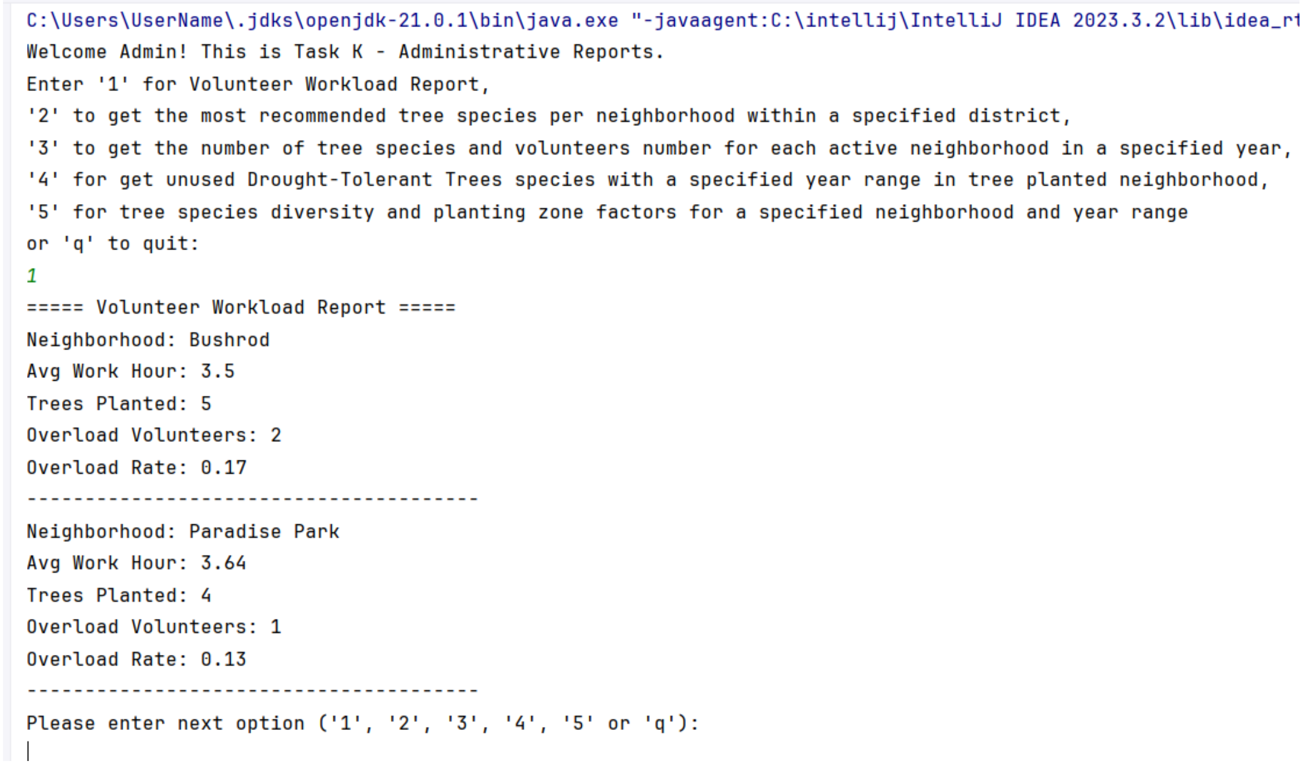
Task J: For every species of trees, find the number of trees planted and some basic statistics on when trees were planted: 1. the number of years since the first tree of the species was planted, 2. the number of years since the most recent tree of the species was planted. 3. In addition, include the year that had the most trees of the species planted and the number of trees planted.



```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_
Welcome to task J, you can see some relevant statistics of trees species planted in Oakland.
Please enter the tree species ID you want to search for:
7
========== Tree Species Planting Statistics ==========
Tree Species ID: 7
Total Trees Planted: 4
Years Since First Planting: 0 years
Years Since Most Recent Planting: 0 years
Year With Most Trees Planted: 2025
Number of Trees Planted in That Year: 4
======================================================
```

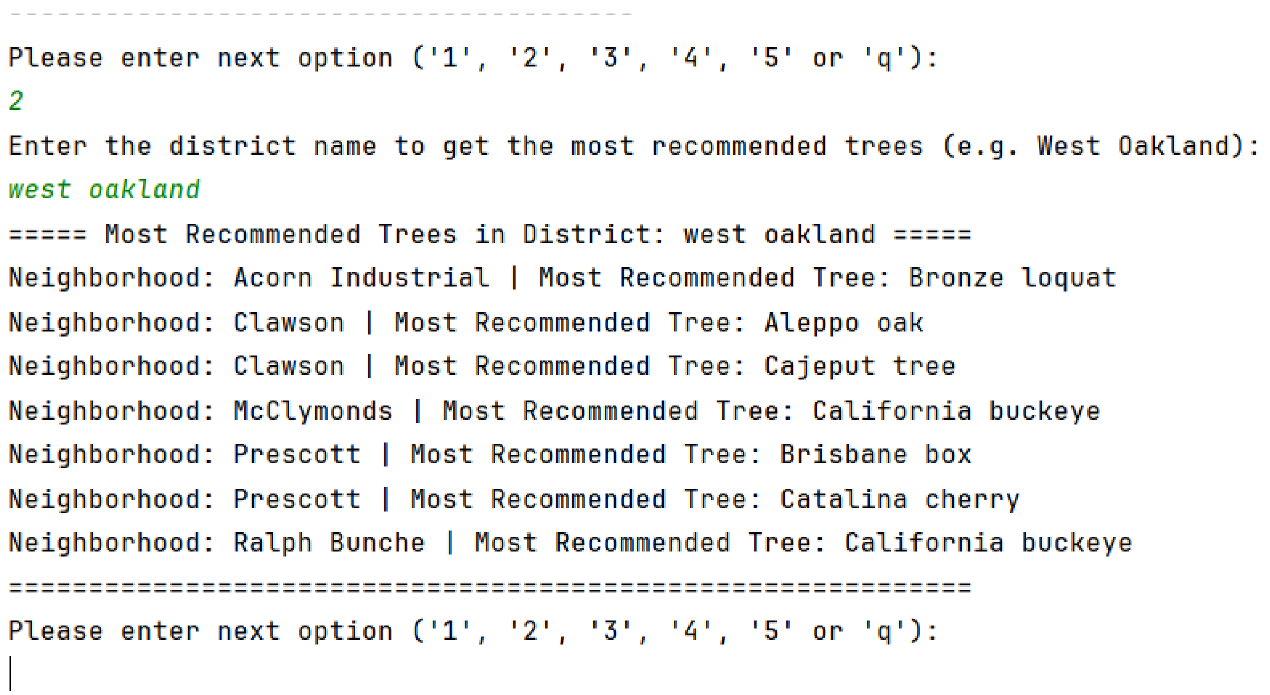
Task K. For each Oakland neighborhood, create a report that summarizes the requests, their progress (pending, in-process, completed, etc.), the trees planted, etc. This is an opportunity for your team to demonstrate your skills, so it's expected that you'll demonstrate sophisticated database querying skills.

K1:



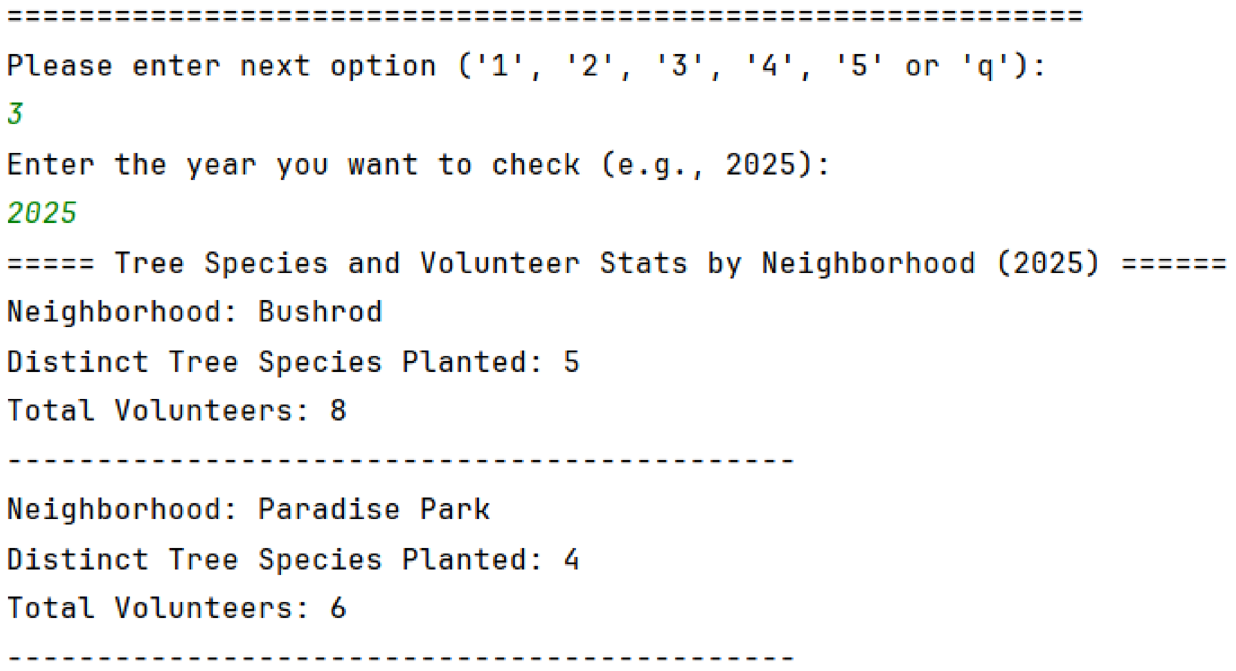
```
C:\Users\UserName\.jdks\openjdk-21.0.1\bin\java.exe "-javaagent:C:\intellij\IntelliJ IDEA 2023.3.2\lib\idea_r
Welcome Admin! This is Task K - Administrative Reports.
Enter '1' for Volunteer Workload Report,
'2' to get the most recommended tree species per neighborhood within a specified district,
'3' to get the number of tree species and volunteers number for each active neighborhood in a specified year,
'4' for get unused Drought-Tolerant Trees species with a specified year range in tree planted neighborhood,
'5' for tree species diversity and planting zone factors for a specified neighborhood and year range
or 'q' to quit:
1
===== Volunteer Workload Report =====
Neighborhood: Bushrod
Avg Work Hour: 3.5
Trees Planted: 5
Overload Volunteers: 2
Overload Rate: 0.17
---------------------------------------
Neighborhood: Paradise Park
Avg Work Hour: 3.64
Trees Planted: 4
Overload Volunteers: 1
Overload Rate: 0.13
---------------------------------------
Please enter next option ('1', '2', '3', '4', '5' or 'q'):
```

K2:



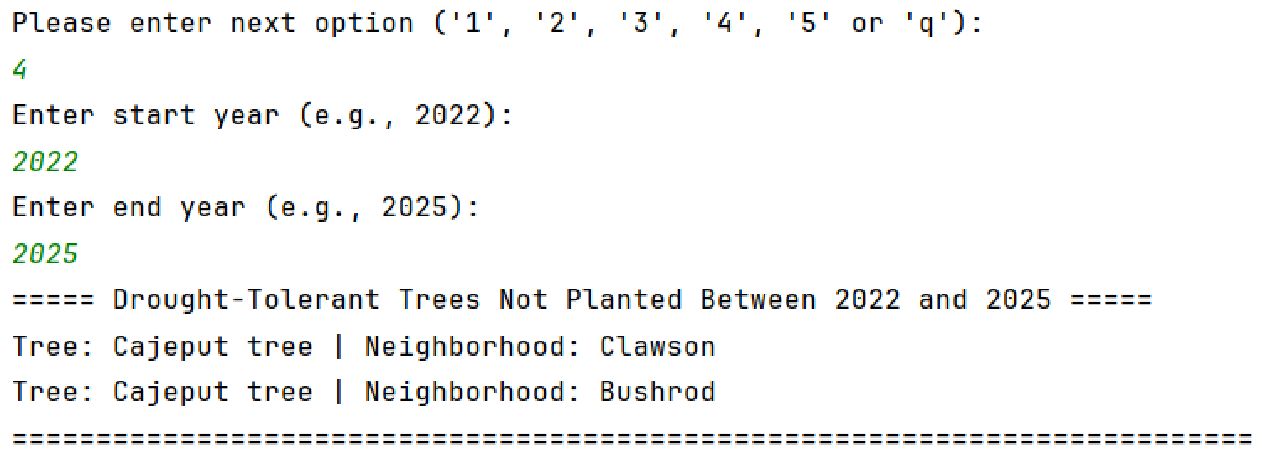
```
---------------------------------------
Please enter next option ('1', '2', '3', '4', '5' or 'q'):
2
Enter the district name to get the most recommended trees (e.g. West Oakland):
west oakland
===== Most Recommended Trees in District: west oakland =====
Neighborhood: Acorn Industrial | Most Recommended Tree: Bronze loquat
Neighborhood: Clawson | Most Recommended Tree: Aleppo oak
Neighborhood: Clawson | Most Recommended Tree: Cajeput tree
Neighborhood: McClymonds | Most Recommended Tree: California buckeye
Neighborhood: Prescott | Most Recommended Tree: Brisbane box
Neighborhood: Prescott | Most Recommended Tree: Catalina cherry
Neighborhood: Ralph Bunche | Most Recommended Tree: California buckeye
============================================================
Please enter next option ('1', '2', '3', '4', '5' or 'q'):
```

K3:



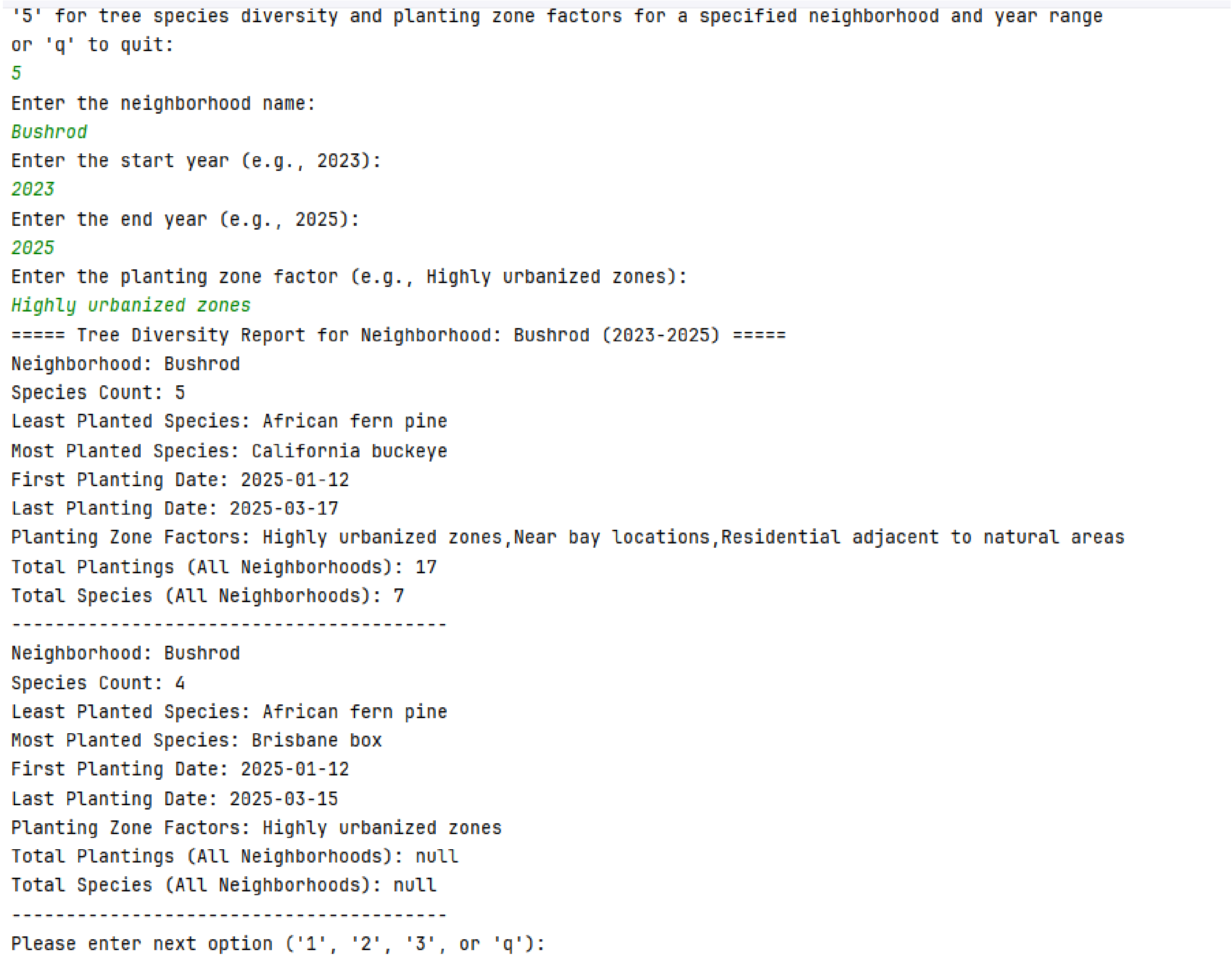
```
============================================================
Please enter next option ('1', '2', '3', '4', '5' or 'q'):
3
Enter the year you want to check (e.g., 2025):
2025
===== Tree Species and Volunteer Stats by Neighborhood (2025) ======
Neighborhood: Bushrod
Distinct Tree Species Planted: 5
Total Volunteers: 8
--------------------------------------------
Neighborhood: Paradise Park
Distinct Tree Species Planted: 4
Total Volunteers: 6
--------------------------------------------
```

K4.



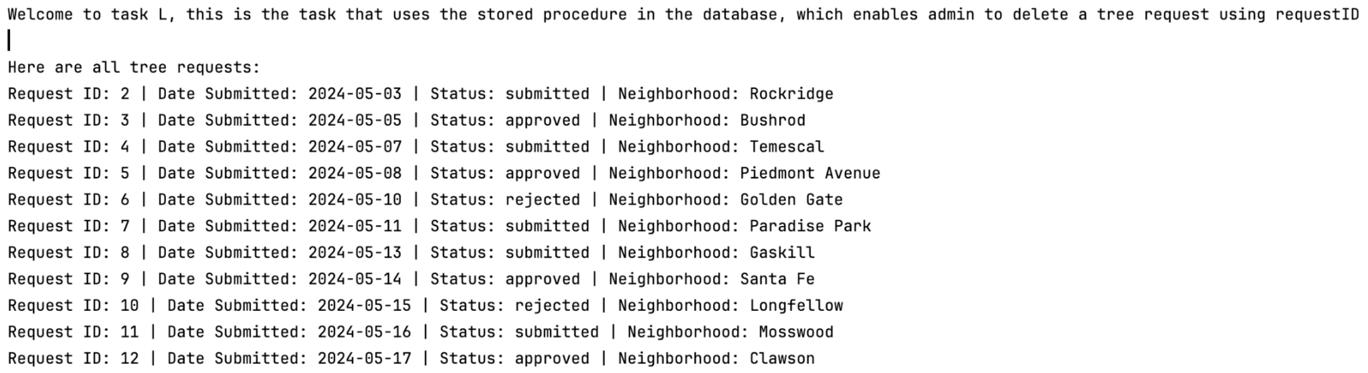
```
Please enter next option ('1', '2', '3', '4', '5' or 'q'):
4
Enter start year (e.g., 2022):
2022
Enter end year (e.g., 2025):
2025
===== Drought-Tolerant Trees Not Planted Between 2022 and 2025 =====
Tree: Cajeput tree | Neighborhood: Clawson
Tree: Cajeput tree | Neighborhood: Bushrod
==========================================================================
```

K5.

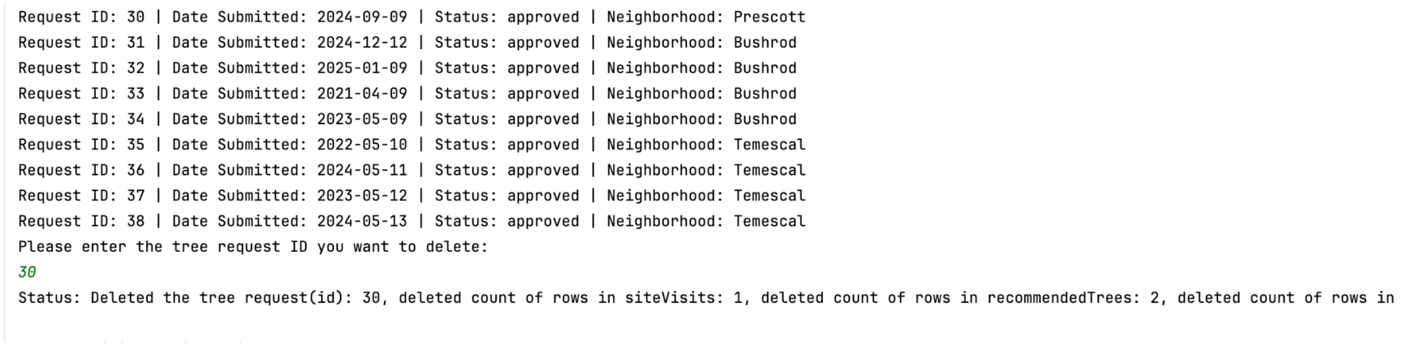


```
'5' for tree species diversity and planting zone factors for a specified neighborhood and year range
or 'q' to quit:
5
Enter the neighborhood name:
Bushrod
Enter the start year (e.g., 2023):
2023
Enter the end year (e.g., 2025):
2025
Enter the planting zone factor (e.g., Highly urbanized zones):
Highly urbanized zones
===== Tree Diversity Report for Neighborhood: Bushrod (2023-2025) =====
Neighborhood: Bushrod
Species Count: 5
Least Planted Species: African fern pine
Most Planted Species: California buckeye
First Planting Date: 2025-01-12
Last Planting Date: 2025-03-17
Planting Zone Factors: Highly urbanized zones,Near bay locations,Residential adjacent to natural areas
Total Plantings (All Neighborhoods): 17
Total Species (All Neighborhoods): 7
----------------------------------------
Neighborhood: Bushrod
Species Count: 4
Least Planted Species: African fern pine
Most Planted Species: Brisbane box
First Planting Date: 2025-01-12
Last Planting Date: 2025-03-15
Planting Zone Factors: Highly urbanized zones
Total Plantings (All Neighborhoods): null
Total Species (All Neighborhoods): null
----------------------------------------
Please enter next option ('1', '2', '3', or 'q'):
```

Task L: This is the task that we defined ourselves, allowing admins to review all the tree requests and delete one using the request id. This is done by implementing a stored procedure in the database. The Java code just calls the stored procedure and all the needed steps are completed within the database.



```
Welcome to task L, this is the task that uses the stored procedure in the database, which enables admin to delete a tree request using requestID

Here are all tree requests:
Request ID: 2 | Date Submitted: 2024-05-03 | Status: submitted | Neighborhood: Rockridge
Request ID: 3 | Date Submitted: 2024-05-05 | Status: approved | Neighborhood: Bushrod
Request ID: 4 | Date Submitted: 2024-05-07 | Status: submitted | Neighborhood: Temescal
Request ID: 5 | Date Submitted: 2024-05-08 | Status: approved | Neighborhood: Piedmont Avenue
Request ID: 6 | Date Submitted: 2024-05-10 | Status: rejected | Neighborhood: Golden Gate
Request ID: 7 | Date Submitted: 2024-05-11 | Status: submitted | Neighborhood: Paradise Park
Request ID: 8 | Date Submitted: 2024-05-13 | Status: submitted | Neighborhood: Gaskill
Request ID: 9 | Date Submitted: 2024-05-14 | Status: approved | Neighborhood: Santa Fe
Request ID: 10 | Date Submitted: 2024-05-15 | Status: rejected | Neighborhood: Longfellow
Request ID: 11 | Date Submitted: 2024-05-16 | Status: submitted | Neighborhood: Mosswood
Request ID: 12 | Date Submitted: 2024-05-17 | Status: approved | Neighborhood: Clawson
```



```
Request ID: 30 | Date Submitted: 2024-09-09 | Status: approved | Neighborhood: Prescott
Request ID: 31 | Date Submitted: 2024-12-12 | Status: approved | Neighborhood: Bushrod
Request ID: 32 | Date Submitted: 2025-01-09 | Status: approved | Neighborhood: Bushrod
Request ID: 33 | Date Submitted: 2021-04-09 | Status: approved | Neighborhood: Bushrod
Request ID: 34 | Date Submitted: 2023-05-09 | Status: approved | Neighborhood: Bushrod
Request ID: 35 | Date Submitted: 2022-05-10 | Status: approved | Neighborhood: Temescal
Request ID: 36 | Date Submitted: 2024-05-11 | Status: approved | Neighborhood: Temescal
Request ID: 37 | Date Submitted: 2023-05-12 | Status: approved | Neighborhood: Temescal
Request ID: 38 | Date Submitted: 2024-05-13 | Status: approved | Neighborhood: Temescal
Please enter the tree request ID you want to delete:
30
Status: Deleted the tree request(id): 30, deleted count of rows in siteVisits: 1, deleted count of rows in recommendedTrees: 2, deleted count of rows in
```

A project retrospective:

1. **UML and Scheme**

   This project provides a practical opportunity to develop an application starting from UML class design, mapping the relations to scheme, creating the database then populating the database. Designing UML class diagrams includes many details such as what attributes should be in the class and how classes are associated with others. This requires understanding of the Tree of Oakland tree requests procedures and constraints, in this case, it requires referring to the steps of filling the tree request form, modeling the characteristics of tree species, checking site visit criterias and understanding how those criterias determine a suitable tree.

   In the process of designing the UML class diagram, we practice using generalization to represent superclass and subclass, using enum data type to represent a set of named, predefined values, such as application status and tree planting zone factors. Since we are using MySQL, all enum data type attributes are represented in lookup tables to maintain flexibility as we can update statuses without altering the main table schema. We also consider if data redundancy is an issue when using zip code attribute in 2 different classes: treeRequest and User, and determine that each zip code attribute could represent different values in those 2 classes and it should be kept. Justifying if using a surrogate key is necessary as other attributes could be used as a primary key, example is that we use requestID as PK in siteVisit scheme as treeRequest associates with siteVisit, and requestID can be a unique identifier, so surrogate key is not necessary in siteVisit scheme.

2. **Java Application Evaluation**

   2. 1. Strengths and What I Liked

   Our Java application satisfies all the required tasks. In particular, for Task K, we successfully implemented five complex SQL queries and for Task L we called a stored procedure.

   There are three aspects of the project that I especially liked:

   - First, thinking through the justification and motivation for each report query helped me better understand how data serves real-world problem solving and provides business insights. This experience made me realize that data is not just for storage but plays an important role in driving decisions.
   - Second, reorganizing the UML diagrams and database schema from a programming perspective was a new experience for me. I had not done this kind of work before. It involved mapping out the workflows for different user roles and defining their permissions clearly. Although, due to time and resource constraints, there were some limitations (which I mentioned below).
   - Third, managing the relationship between real-world workflows and application logic gave me a deeper understanding of system design, even though the implementation wasn't perfect.

3. **What we found easiest and hardest**

   a. **Easiest part**

Drawing the UML diagrams and creating the database schema felt relatively easy once we were clear about the classes and attributes, since we had been practicing them throughout the semester.

b. **Hardest part**

It was challenging to map out workflows for different user roles and define their permissions clearly. Also, managing the relationship between real-world workflows and application logic was tough, even though it helped me better understand system design.

4. **What we learned**

The most impressive thing we learned is how smooth it is when writing SQL statements for the database by referring to the relational scheme we have. Not only for the query part(which we did for the assignments), but also for ddl that defines the tables and constraints. Everything has a corresponding part that is already demonstrated in the scheme, allowing us to check for the correctness.

Also, the progress of designing and iterating our database which is tightly connected to a real world problem is really meaningful. Real world scenario enables us to come up with a basic expectation of what our database should be modeling, and with the deep dive of our tasks, we gradually find out what is important for the application and what is not, what is efficient and what is redundant. This gives us a valuable experience that can be applied to our career as a software development engineer.

And speaking of the RA, execution plan and stored procedure, this gives us a new aspect of thinking about our design. For most of our work we first only focus on functionality, but it should be kept in mind that efficiency and sustainability are really important, and as we keep working on the relational algebra we think of alternative ways to achieve one functionality, the execution plan gives us a more straightforward way to see whether our SQL queries work efficiently. And the stored procedure is something that we can utilize for reducing multiple data transfer through the network for only one task.

Conclusion:

We have completed the following tasks:

- Design and draw the UML class diagram and map it to the scheme
- Come up with the proper SQL statements that set up the database and initialize the data
- Fulfilled the following tasks by using Java code:
    - Registered new users: volunteers, residents and admin.
    - Processed tree requests, including collecting the planting address, planting permit status.
    - Scheduled site visits for tree requests and assigned admin members.
    - Recorded site visit details, including photos for team review.
    - Scheduled tree planting and assigned volunteer teams.
    - Documented volunteer participation, site observations, and before/after photos after planting.
    - Updated the available tree species list by adding or removing species.
    - Displayed the status and age of all pending or in-progress tree requests.

    - Enabled filtering to find trees planted in selected Oakland neighborhoods or zip codes.
    - Provided species specific report, including:
        - Total trees planted per species
        - Years since the first and most recent planting
        - The peak year and total for most trees planted.
- Generated neighborhood level reports summarizing:
    - Tree request statuses (submitted, approved, rejected)
    - Trees species and amount planted
    - Generate volunteers workload report
    - Generate the most recommended trees in district report
    - Generate the tree species and volunteers statistics report
    - Generate a report to show drought tolerant trees not planted in specific range of year
    - Generate a report to analyzes tree species diversity and planting zone factors for a specified neighborhood and year range
- Implemented a stored procedure allowing admin to delete a tree request by entering the request id.

What still needs to be done:

- For designing part:
    - Images:
        - It might be better that things like images should have a separate UML class, because for the site visit and tree planting there might be multiple photos that needs to record, however, in our model we choose not to do so because this is kind of 'off the topic' since we only have command lines to demonstrate the input and outputs, which is hard to show the images.
    - Admins:
        - For the design of different roles of the user, it might be more reasonable that admin has its own attributes that makes it a subclass in the UML diagram, for example, admin may be responsible for some part of Oakland and only deal with those area's tree request and volunteer booking tasks, which leads to an attribute like 'responsible area'.
- For queries:
    - It would be nice to have some views that store the temporary result that is used multiple times for a query or frequently being queried.
- For the Java code, there are many places that can have a better code quality, here are some of the points we noticed:
    - Lack of object-oriented design in some parts:
        - Due to the tight timeline and limited team size, some parts of the code mix DAO logic and UML class logic. Some UML classes were not fully utilized, and the application focused mainly on "passing data" instead of enforcing clear object-oriented principles.
    - Lack of unit testing and defensive programming:

        - We did not have enough time to properly implement unit tests. Many methods did not fully follow defensive coding practices. For example, although we made user input case-insensitive in some cases, edge case handling overall was not comprehensive.
    - Access modifiers and method visibility were not carefully designed:
        - For the sake of time, we did not carefully distinguish which getter/setter methods should be public or private. In a real application, method visibility should be carefully considered to improve security and maintainability.
    - Manual tree species assignment:
        - Currently, the admin manually assigns a tree species by ID. Although we collected enough data (such as planting zones and tree characteristics) to support automatic recommendation, this functionality was not implemented yet.
    - Limited input validation and exception handling :
        - In some areas, especially during user input and database operations, we could improve exception handling and prevent invalid states (e.g., invalid dates, non-existing IDs).
    - Scalability and optimization were not considered :
        - Some SQL queries might become slow if the dataset grows larger. In a real-world deployment, indexing and query optimization would be necessary.

In future development, we can improve the application in the following ways:

- Refactor the codebase to better separate concerns between DAO classes, service classes, and entity classes, ensuring cleaner object-oriented design.
- Add unit testing for all major functions to catch bugs early and ensure better code quality.
- Improve input validation and exception handling to deal with edge cases and invalid user input more gracefully.
- Implement automatic tree species recommendation based on factors like planting zone, drought tolerance, and neighborhood characteristics.
- Review and correct method visibility (private/public) and apply best security practices.
- Optimize SQL queries and database indexes to improve performance and prepare for larger datasets.
- Adopt a modern framework like Spring Boot to improve application usability, scalability, and maintainability. It also allows for better dependency management, cleaner REST API development, and easier deployment in production environments.

Individual contribution statements：

Yunyu Guo:
For milestone:

1. Contribute to design UML class diagram
2. Address the classes design assumptions
3. Modify the scheme and checking relationships between keys
4. Initial completion of tasks a to k using stored procedures and sql queries
5. Wrote a complex report query

For final project:

1. Redesign UML class diagram
2. Modify the relation scheme
3. Write justifications of attributes regarding duplication, 1 to many relations in user and treeRequest, justify the use of surrogate key of plantID, and relevant assumptions
4. Draft the final report
5. Complete a complex query report using parameters
6. Draft presentation slides
7. Write 2 relational algebra expressions and its equivalent form
8. Testing Java code and review teammates' report queries
9. Record the annotated walkthrough video
10. Draft the README.md

Siyuan Liu:

For milestone:

1. Drafted the initial design document, including analysing the class need to be modeled, attributes that might be useful
2. Actively engaged in all the discussion for all team meeting
3. Constantly update the UML with others
4. Updated scheme diagram
5. ddl: wrote users, volunteers, neighborhoods, permits, treeRequests, treePlantings, siteVisits
6. dml: insert data for users, volunteers, neighborhoods, permits, treeRequests, treePlantings, siteVisits
7. reports: wrote 2 complex query

For final submission:

1. Modify the UML class diagram
2. Modify the relation scheme
3. Complete half the refactor work on ddl.sql and most work on dml.sql to ensure that we can use these two files to set up our oaktrees database with reasonable data
4. Help with the final report (abstract, problem definition, checking for justifications, 3NF, etc.)
5. Initial completion of the SQL queries used for 2 complex reports, including:
    a. create a report about the volunteer average workload for each neighborhood that is in the highest average work hour district to help analyse whether we need more volunteers for those neighborhood
    b. a report about the most recommended tree species(common name) for all the neighborhood in the chosen district by user input to help the admin to decide whether some tree species need more inventory
6. Completion of the relation algebra and its equivalent RA of 2 complex queries
7. Analysing the execution plan and come up with a visualization of it
8. Completion of the stored procedure of deleting a tree request and the related Java code
9. Help review & testing the Java application

Ting Li:
For milestone:

1. Actively engaged all the discussion for all team meeting.
2. Drafted the description file, and others reviewed and gave suggestions.
3. Drew the UML initially, and constantly update it with others.
4. Updated scheme diagram sometimes.
5. ddl: wrote plantingZones, siteVisits, trees, and recommendedTrees.
6. dml: insert data for plantingZones, siteVisits, trees, and recommendedTrees.
7. reports: wrote 2 complex query.
8. wrote repo.txt with markdown file draft.
9. pre-release quality check and pre-release v-0.5.

For Final Submission

1. Design and implement and debug and test all java applications.
2. Design UML class diagram
3. Modify the relation scheme
4. Help with the final report, including Screenshots of the running system, and finish part of 2. Java Application Evaluation, 3. What we found easiest and hardest, and Java part in What still needs to be done, and future recommendation of spring boot.
5. Write project retrospective of section 2. Java application evaluation.
6. Initial completion of the SQL queries used for 2 complex reports, including:
    a. track tree species and volunteer number impact in actively planting neighborhood in a given year.
    b. get unused Drought-Tolerant Trees species with a specified year range in tree planted neighborhood.
7. Recording the java application demo video.

Reference

https://www.oaklandca.gov/resources/official-street-tree-list

https://www.treesforoakland.org/how-to-get-a-tree.html

https://docs.google.com/forms/d/e/1FAIpQLSccg6WxSEgPx-epsUc4OKMJuGkzMUuCtZP2xw1nyW-6p-7kug/viewform

https://localwiki.org/oakland/Neighborhoods

https://sqldatagenerator.com/generator

https://docs.oracle.com/javase/8/docs/api/java/sql/Statement.html