Roman Carrubba, Ricardo de Guzman, Arjun Sikka

CS3200 Spring 2025

April 21, 2025

Abstract

Trees for Oakland, a non-profit organization committed to urban greening, currently uses manual processes and Google Forms to manage street tree planting requests. This approach makes it difficult to track the status of tree requests, coordinate volunteers, monitor tree species inventory, and generate data. The lack of a centralized system obstructs efficiency, communication, and transparency between the organization and Oakland residents.

To resolve the issue, OAKhoury Trees is a database-backed application designed to streamline the entire tree planting workflow. The system allows residents to create accounts, submit requests for tree planting, and apply for permits. Volunteers can register and be assigned to tree planting events. Staff can track and update each stage of the process from request submissions to site visits then final tree plantings. The platform will also maintain an updated tree species inventory, manage volunteer engagement, and provide analytical reports to support planting and community impact measurement.

The application includes secure user registration and role-based access, request submission with street-level detail, permit tracking, and status updates. It enables staff to schedule site visits, upload photos, assign volunteers, track planting progress, manage tree species inventory, and generate reports analyzing trends by species, neighborhood, and planting activity over time.

This solution will replace the current inefficient system with a robust, normalized relational database and user-friendly application. It enables better service delivery, fosters community involvement, and provides valuable data driven insights for both internal operations and external reporting. OAKhoury Trees supports a scalable and maintainable approach to urban environmental administration.

Entities Involved

- User represents all users of the system, including residents, volunteers, and team members. Its attributes include firstName, lastName, email, neighborhood, zipCode, and isVolunteer. Users that are residents can submit tree requests, participate in planting events as volunteers, or manage site visits and planting activities as a team member.
- TreeRequest is an association class that tracks requests submitted by residents for tree planting at specific addresses. Its attributes include treeType, payment, address, and status (Enum: pending, approved, or planted). Each request is linked to a specific user and progresses through various stages (permit application, site visit, & planting).
- JobSite represents the physical location where trees are planted. Its attributes include address, zipCode, name, and description. Each job site is associated with a neighborhood for reporting purposes.
- Neighborhood is an enum class that denotes the Oakland neighborhood where planting occurs. It has a single attribute name, used for linking job sites for reporting purposes.
- TreeSpecies contains information about city-approved tree species available for planting. Its attributes include species characteristics such as name, minimumBasinWidth, commonName, scientificName, heightRange, acceptableUnderPowerLine, droughtTolerance, foliage, rootDamagePotential, and numberPlanted.
- SiteVisit is an association class that represents inspections conducted by team members at job sites before planting approval. Its attributes include scheduledDate, conductedDate and photoURL.
- PlantingEvent is an association class that tracks events where trees are planted by volunteers under team member supervision. Its attributes include scheduledDate, completedDate, member, eventLeader, observation, and photoURL.
- Employee represents team members who manage site visits and lead planting events. Its attributes include id and salary.
- Role is an Enum indicating the role assigned to an employee.

Relationships Between Entities

- User requests a TreeRequest. A user can submit multiple tree requests, but each request belongs to only one user (one-to-many).
- TreeRequest is requested to a JobSite. Each tree request is associated with one job site where the tree will be planted (many-to-one).
- JobSite is located in Neighborhood. Each job site is located within one neighborhood, but a neighborhood can have multiple job sites (many-to-one).

- JobSite has a SiteVisit. Each job site can be visited multiple times by employees for inspection (one-to-many)
- JobSite has a PlantingEvent. Multiple planting events can occur at a single job site over time (one-to-many).
- PlantingEvent plants TreeSpecies. A single planting event may involve planting multiple trees of one species, but each tree belongs to only one event (many-to-one).
- Employee conducts the SiteVisit. Multiple employees may participate in or manage site visits at different job sites (one-to-many).
- Employee manages JobSite. Employees can manage multiple job sites and each site can have multiple managers. (many-to-many)
- User volunteers at PlantingEvent. Volunteers participate in multiple planting events, but each event involves multiple volunteers (many-to-many).

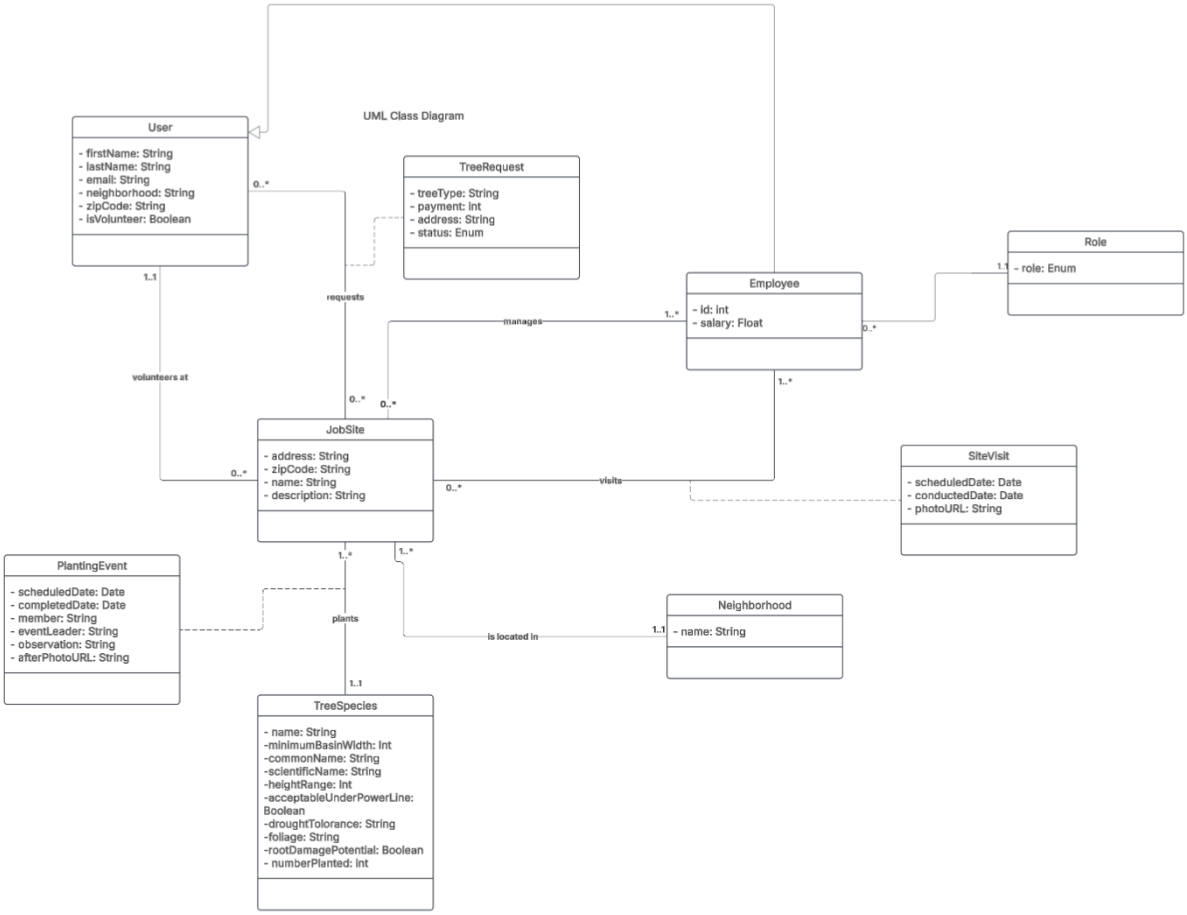
Assumptions

- Users must register before submitting a tree request or volunteering for an event.
- Neighborhood data is pre-defined based on Oakland LocalWiki information.
- Photos are stored as URLs pointing to external storage systems.
- Employees inherit from users via foreign key lineage
- Role and status are represented as Enum types for consistency and validation

Information to Be Extracted/Presented

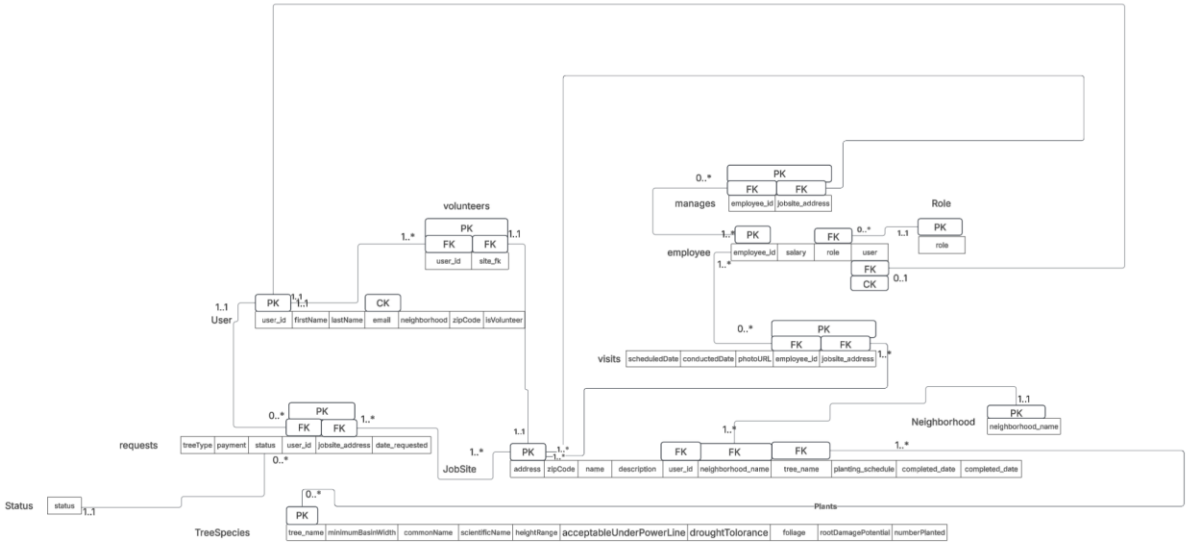
- User information that includes details of registered users, including their roles (resident, volunteer, or team member) and associated actions like submitted requests or participation in events.
- Tree requests statuses of all tree requests (*pending*, *approved*, *planted*, etc.) along with timestamps for tracking progress that's filtered by neighborhoods or zip codes.
- Site visit dates for visits that are scheduled and conducted and photos taken during visits.
- Planting events details including dates, participating volunteers and supervisors and number of trees planted during each event.
- Neighborhood reports that include the summary of requests submitted per neighborhood, progress of requests, and total number of trees planted in each neighborhood.
- Species statistics that include the number of trees planted per species and historical data for each species such as years since the first tree was planted, since the most recent tree was planted, and year with the highest number of plantings.
- List of volunteers who participated in specific events and their roles.

UML Class Diagram




[UML Class Diagram LucidChart]

RDB Scheme Diagram




[RDB Scheme Lucidchart]

Normalized Relations

- All the relations are in 3rd Normal form
- We assumed that when a user requests a job site, the job site is in the database immediately, and there will be a few null values. This means that requests are a junction table with User and Job site foreign keys. Status is an Enum variable of requests as well, so it branches off and has a 1 to M relationship.
- A User can also volunteer at many job sites. So, a volunteer junction table was created for volunteers with the job site and user foreign keys. This makes it very efficient to query to find users and what job sites they volunteered for.
- The user has a surrogate key to save memory because it gets copied around a lot.
- We decided not to use a surrogate key for JobSite and keep it as the address to make queries easier and more efficient.
- The tree species table only has one foreign key relating to it, so we decided not to have a surrogate key. This decision was also based on the website where they have a table of every tree species and all the attributes in our database, and they didn't have a surrogate key, so this was part of our decision not to have a surrogate key.
- We made the assumption that pictures before the tree planting would be taken by an employee when they visit and employees can visit many sites we decided that sites could possibly have multiple visits so we decided to make a junction table that also had a photo of the job site before the tree was planted as that is when the photo would be taken.
- We made the assumption that multiple employees could oversee a single job site, which means that there is a many-to-many relationship between the job site and the employee.
- The employee has a surrogate key to save on memory because of how many foreign keys relate back to employee. In addition, the role attribute of employee is an Enum value, so it branches off and has a 1 to M relationship. The user attribute of employee also serves as an FK and CK because of its inheritance with the user class.
- While the neighborhood only has one foreign key relating to it, the neighborhood strings were very long, so we decided to give it a surrogate key to save on space and to lessen the chance of someone making a typo when inputting the Neighborhood ID in Job Site. Neighborhood name is an Enum variable of jobsite as well, so it branches off and has a 1 to M relationship.

Screenshots

Task 1: add a user

```
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
3
Enter fist name: roman
Enter last name: carrubba
Enter email: rcarrubba@gmail.com
Enter neighborhood: Downtown
Enter zipCode: 94110
User added successfully!
Your ID number is 75507
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Remove a tree
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
```

Task 2: accept a request

```
Please choose from the following options
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Remove a tree
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
4
Enter address: 555 Spruce St
Tree request accepted!
```

Task 3: Schedule a visit

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Remove a tree
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
5
Enter Address:
555 Spruce St
Enter Scheduled Date:
 Please format as: yyyy-mm-dd
2025-03-02
Visit scheduled
```

Task 4: record information for a visit

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Remove a tree
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
6
Enter Address:
555 Spruce St
Enter conducted date:
 Please format as: yyyy-mm-dd
2025-04-20
Enter Photo URL: https://www.testexample.com/pleasegivemeagoodgrade
Visit Recorded
```

Task 5: Schedule the planting of a tree

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Remove a tree
9: Add a tree
10: View pending requests
11: Find tree locations
12: Get tree statistics
13: Get neighborhood statistics
13: Exit
7
Enter address of job site: 555 Spruce St
Enter planting date (format yyyy-mm-dd): 2025-06-12
Planting date scheduled!
Enter volunteer user IDs to assign to this site (comma-separated):
1,2
Volunteers successfully assigned!
```

Task 6: Record planting event

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Record a planting event
9: Remove a tree
10: Add a tree
11: View pending requests
12: Find tree locations
13: Get tree statistics
14: Get neighborhood statistics
15: Exit
6
Enter Address:
555 spruce St
Enter conducted date:
 Please format as: yyyy-mm-dd
2025-11-01
Enter Photo URL: exampleURl
Visit Recorded
```

Task 7: Update the collection of tree species

Addition:

```
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Record a planting event
9: Remove a tree
10: Add a tree
11: View pending requests
12: Find tree locations
13: Get tree statistics
14: Get neighborhood statistics
15: Exit
10
Enter tree species details to add a new species:
Tree species name: red wood
Minimum basin width: 3
Common name: rew wood
Scientific name: Sequoia sempervirens
Height range (e.g., 40-70 ft): 70-100 ft
Acceptable under power lines (TRUE/FALSE): FALSE
Drought tolerance (e.g., Low, Medium, High): Low
Foliage (e.g., Deciduous, Evergreen): Evergreen
Root damage potential (e.g., Low, Medium, High): High
Number of trees planted: 150
Years since first tree: 15
Years since most recent tree: 2
Year with most trees planted: 2023
New tree species added successfully!
```

Removal:

```
Please choose from the following options
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Record a planting event
9: Remove a tree
10: Add a tree
11: View pending requests
12: Find tree locations
13: Get tree statistics
14: Get neighborhood statistics
15: Exit
9
Enter tree name for deletion:red wood
Tree deleted
```

Task 8: show all non completed request

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Record a planting event
9: Remove a tree
10: Add a tree
11: View pending requests
12: Find tree locations
13: Get tree statistics
14: Get neighborhood statistics
15: Exit
11
Pending Tree Planting Requests:
Status: Pending, Days since submitted: 794
Status: Approved, Days since submitted: 771
Status: In Progress, Days since submitted: 839
Status: Delayed, Days since submitted: 799
Status: Cancelled, Days since submitted: 780
Status: Scheduled, Days since submitted: 766
Status: On Hold, Days since submitted: 756
Status: Submitted, Days since submitted: 749
Status: Finalized, Days since submitted: 730
```

Task 9:

```
Please choose from the following options:
1: Log in as a user
2: Log in as a employee
3: Sign up as a user
4: Accept a tree request
5: Schedule a visit
6: Record a visit
7: Schedule a tree planting
8: Record a planting event
9: Remove a tree
10: Add a tree
11: View pending requests
12: Find tree locations
13: Get tree statistics
14: Get neighborhood statistics
15: Exit
12
Would you like to search by zip code or neighborhood?
1: Zip Code
2: Neighborhood
2
Enter Neighborhood: DownTown
Tree species count for the selected location:
Tree: Red Maple - Count: 1
```

Task 10: Find all trees planted within a selection of Oakland neighborhoods(NOTE: not all of the results fit in one screen shot, so this is a small portion of the results)

```
15: Exit
13
Tree Species: Valley Oak
Total Trees Planted: 200
Years Since First Planted: 2015
Years Since Most Recent Planted: 2023
Year with Most Planted: 2021
Most Planted Year Count: 200
------------
Tree Species: White Ash
Total Trees Planted: 170
Years Since First Planted: 2019
Years Since Most Recent Planted: 2023
Year with Most Planted: 2021
Most Planted Year Count: 170
------------
Tree Species: Monterey Pine
Total Trees Planted: 150
Years Since First Planted: 2017
Years Since Most Recent Planted: 2024
Year with Most Planted: 2022
Most Planted Year Count: 150
------------
Tree Species: rew wood
Total Trees Planted: 150
Years Since First Planted: 2010
Years Since Most Recent Planted: 2023
Year with Most Planted: 2023
```

Task 11:Neighboorhood reports(NOTE: not all of the results fit in one screen shot, so this is a small portion of the results)

```
 14
Neighborhood: Bayview
Total Requests: 1
Pending Requests: 0
In-Process Requests: 0
Completed Requests: 1
Approved Requests: 0
Total Trees Planted: 150
Trees Planted: Monterey Pine
------------
Neighborhood: Castro
Total Requests: 1
Pending Requests: 0
In-Process Requests: 0
Completed Requests: 0
Approved Requests: 0
Total Trees Planted: 140
Trees Planted: Weeping Willow
------------
Neighborhood: Chinatown
Total Requests: 1
Pending Requests: 0
In-Process Requests: 0
Completed Requests: 0
Approved Requests: 0
Total Trees Planted: 110
Trees Planted: American Elm
```

Project Retrospective

Our team was able to create with every stage of a relational database system thanks to this database application project. As we reflected on the experience, we pinpointed various elements that we appreciated, aspects that posed difficulties for us, and lessons learned that will shape our future endeavors.

What we liked the most:

Our favorite part of the project was designing the UML class diagram and writing the SQL queries. These tasks allowed us to think critically about the structure of our data and how best to retrieve meaningful information. Crafting the UML model gave us a clear high-level view of our system and set a solid foundation for everything that followed. It functioned as a great segway to mapping the RDB and writing the SQL code. The SQL phase felt rewarding, especially when complex queries returned exactly the insights we expected.

What we disliked the most and found the most difficult:

The most difficult and least enjoyable part of the project was building the Java-based application interface. While we were able to fulfill the requirements successfully, implementing a user-friendly and functional interface was time-consuming and riddled with problems we had to overcome. Despite the difficulties, this experience gave us a good challenge and taught us how to integrate database work with a command line interface.

What we found easiest:

The easiest part of the project was mapping our UML class diagram to the relational database schema. Once the UML model was complete, converting it into an RDB and setting up foreign key relationships was pretty straightforward. The clarity of our class diagram made this transition relatively seamless. It also helped knowing that in the end our RDB had to be in 3NF which was in the back of our minds when creating and reviewing the final product.

What we learned the most:

The biggest takeaway from this project was understanding how each component of a database system connects and builds upon the last. We learned how important it is to approach design decisions thoughtfully from the very beginning, as early choices in our UML model had a direct impact on our relational schema, queries, and final application. We also gained hands-on experience with writing complex SQL queries and managing data, which strengthened our grasp of how databases operate under the hood. Overall,

this project reinforced the value of planning, testing thoroughly, and working collaboratively through each phase of development.

Conclusion Statement

In conclusion, our group fulfilled all requirements presented for this assignment and believe there is no more to be done with respect to the provided instructions. Starting with a fully fledged UML class diagram, we were able to map it to a RDB schema to present relationships between different entities. Transferring the RDB schema to SQL code allowed us to pursue the desired and additional tasks needed. Finally, wrapping up the project with the user-based application using Java took our functionality to a new level, where one can interact with the data in real-time. One way this project could go above and beyond is if we implemented a GUI for the application to make it even more interactive and easier for a user to navigate around.

Contribution statements

Roman Carrubba:

We all helped with a bit of everything. However, I contributed most to the tasks and the application.  I also did a portion of the presentation. Specifically the application part of the presentation, which I edited the slides for and presented.  Finally I also helped a lot with the application video, This made the most sense for me to do because I was a large contributor to the application.

Arjun Sikka:

We all helped with a bit of everything. Specifically, I worked on creating the relational schema and implementing suggestions provided to us in the milestone to the schema and the UML diagram. I also helped write the justifications for any decisions we took in creating both of these. I also wrote the project retrospective and conclusion statement found in report.pdf.

Ricardo de Guzman:

We all helped with a bit of everything. Specifically, I worked on creating the UML diagram and initiating the work. I also helped create the description and report queries. Additionally, I created the foundation of the report including its cover sheet, abstract, and description of the problem. I also created the presentation slides and YouTube video.