

# Code Management

All code and documentation can be found on our github <https://github.com/Irisvella/SOEN-390-W2024> . Below, you will find descriptions of our approaches and some examples of the different types of code management techniques undertaken during this project. If any further information is needed, everything can be found in the github provided above.

## **Quality of source code reviews**

By working in teams, multiple people go over the same code to heighten the quality of the overall code. Our team is divided into teams which fluctuate in members throughout different sprints, as well as during sprints if people have finished their tasks, or reprioritization of features takes place. In light of this, subteam meetings happen a few times per sprint to go over internal goals and to make sure the code is of good quality, as well as clear of bugs. If bugs are found they can be logged and discussed.

## **Correct use of design patterns**

In designing our website, app, and database, we focused on several design principles:

- We followed a composite pattern when building our structure by splitting our users into types such as “employee” or “management company” and interconnecting them.
- The decorator pattern is utilized by giving different roles, types and other attributes to our entities, providing them with more depth and behaviors.
- There are some observer patterns used when looking at auto increments and decrements that are incorporated into our database structure. Their activators can be said to observe the system and trigger increments/decrements based on events.
- Using an SQL database like us inherently invokes the repository pattern since our data is stored in tables. Each table can store and manage specific types of data.
- During the creation of different types of users, we can say the factory pattern is in play since we are creating types of objects based on a larger common type “user”.

## Respect to code conventions

As we are using React, we use similar coding conventions to Javascript. As part of this, we respect camelCase for identifier names, leave spaces around operators to improve code readability, indent our code, and follow object rules. Comments are also added before methods.

```
// Route to handle the submission of a new listing
router.post(
  "/",
  verifyToken,
  async function (req: Request, res: Response, next: NextFunction) {
    try {
      jwt.verify(
        req.token as string,
        process.env.SECRET as jwt.Secret,
        async (err, decoded) => {
          if (err) {
            return res.status(401).json("Unauthorized");
          } else {
            console.log("decoded ---- ", decoded);
            const { id, role, email } = (<any>decoded).data;

            if (role === "company") {
              const body = req.body; //constant

              async function createProperty(address: string) {
                const property = await prisma.property.create({
                  data: {
                    //address on the table to the left
                    //address on the body is to the right
                    address: address,
                    size: 0,
                    condo_fee: 0,
                    unit_id: 0,
                  },
                });
              };
              await new Promise(f => setTimeout(f, 1000)); // wait so that the database has time to update the added property
            }
          }
        }
      );
    }
  }
);
```

## Design quality

Design quality can be measured by many metrics and some of them will be seen below. The main metrics I want to highlight here are Readability, Modularity, Consistency, Scalability, Testability, Documentation and Version Control.

Clear naming conventions and consistent formatting were used to achieve readability by members within our team, but also by outside developers. This is especially important as our repository will remain active for others to reference and learn from.

Modularity was also heavily implemented by the use of components. By building pieces of our site in components, we can both reuse code efficiently (for example the navbar) and cleanly. This also allows for much more manageable code blocks, which can be maintained efficiently.

By having early meetings and laying down rules for our coding style and code conventions, we were able to keep a fairly consistent style across our code. It also helped that we chose our modular, component based coding design. Again, this modularity makes maintaining code and conventions easier.

Scalability was kept in mind when designing our database, our design concept, and our choice of languages. Our database was built with the idea that it could always be scaled up, our modularity allows for simple reuse of components to make expanding the site easy, and our architecture choices leave us some flexibility to grow.

As we are designing an application page by page, which are made of smaller components, testing each one is far simpler than having to test something much larger and more imposing. This ease of testing promotes reliability and reduces the likelihood of bugs passing unnoticed.

Documentation can be seen abundantly out of the code with heavy documentation covering all aspects of the project and offering insight to our intentions, as well as decisions. It can also be found inside the code with proper inline comments, an organized README file and good document title comments.

Finally, Version Control is achieved through the use of Git to constantly moderate our codebase. Not only does it offer the ability to track changes in our project, but it improves our collaboration immensely and protects the codebase from harmful changes being made before it can be checked. In case anything harmful does occur, Git allows for rolling back to previous versions, giving even more safety.

## **Quality of source code documentation**

We followed three types of documentation conventions: file level comments, component comments and function comments. File level comments help guide the reader and explain what to expect, who wrote this code and any dependencies. Component comments are similar, but are more precise in the role of this component in the larger scheme of things. Function comments are more atomic explanations of how the function operates. As seen in the examples below, these practices are respected.

```

// Filename: addEmployee.jsx
// Author: Sarah Abellard, Samuel Collette, Abisan
// Description: Component for adding new employees to the system.
// Dependencies: React, MUI (Material-UI)
import React, { useState } from 'react';
import Navbar from '../components/Navbar';
import { Box, Typography, TextField, Button, Select, MenuItem, FormControl, InputLabel } from '@mui/material';

const AddEmployee = () => {
  // State for employee data
  const [employeeData, setEmployeeData] = useState({
    email: '',
    role: '',
  });

  // Handle input change
  const handleChange = (e) => {
    const { name, value } = e.target;
    setEmployeeData((prevData) => ({
      ...prevData,
      [name]: value,
    }));
  };

  // Submit form data
  const handleSubmit = async () => {
    const token = localStorage.getItem('token');
    try {
      const response = await fetch('http://localhost:3000/add-employee', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          Authorization: `Bearer ${token}`,
        },
        body: JSON.stringify(employeeData),
      });

      if (response.ok) {
        console.log('Employee data submitted successfully');
        // Optionally, reset the form
        setEmployeeData({
          email: '',
          employee_id: '',
          company_id: '',
        });
      }
    } catch (error) {
      console.error('Error submitting employee data:', error);
    }
  };

  return (
    <div>
      <Navbar />
      <div>
        <TextField
          type="text"
          name="email"
          value={employeeData.email}
          onChange={handleChange}
          placeholder="Email"
        />
        <TextField
          type="text"
          name="role"
          value={employeeData.role}
          onChange={handleChange}
          placeholder="Role"
        />
        <Button type="button" onClick={handleSubmit}>Add Employee</Button>
      </div>
    </div>
  );
};

export default AddEmployee;

```

## Refactoring activity documented in commit messages

Refactoring was mostly present in the db-changes branch of our project where changes have been made to our database schema as the project has progressed. Many more of these refactors can be found among the commits we have made but they are across all branches. As any other commit, proper naming and descriptions for commits can help us find them among all the commits. Below is an example from the db-changes branch.

add db schemas for operating_fee	bt919 committed 3 days ago	2a3fb0f		
Merge pull request #59 from Irisvella/db-changes	Irisvella committed 3 days ago · ✓ 2 / 2	Verified a79a7a4		
add not null constraints to db	bt919 committed 3 days ago · ✓ 2 / 2	01d04fd		
fix createEditListing to fit new db	bt919 committed 3 days ago · ✗ 0 / 2	fce367a		
add db support for condo units	bt919 committed 3 days ago · ✗ 0 / 2	08a5dfa		
removed depreciated "username" attribute	Irisvella committed 3 days ago · ✓ 2 / 2	bad6ae4		
Revert "Merge branch 'main' into db-changes"	Irisvella committed 3 days ago · ✗ 1 / 2	21b8ab1		
Merge branch 'main' into db-changes	Irisvella committed 3 days ago · ✗ 0 / 2	Verified 3ad614b		
Fixed Navbar and profile dash	Irisvella committed 3 days ago	3687a86		

## Quality/detail of commit messages

Each commit is given a brief, specific message to better track commit history and explain the changes made during this specific commit. Along with the use of many, small commits, we can see the progression of any given feature/branch before it is merged into our main branch.

# db changes and billing route #69

[Edit](#) [Code](#)

Merged

sarahabellard merged 9 commits into `main` from `db-changes` 4 hours ago

Conversation 1

Commits 9

Checks 2

Files changed 4

+392 -68

Commits on Mar 17, 2024

add db schemas for operating\_fee

bt919 committed 2 days ago

2a3fb0f

<>

Commits on Mar 18, 2024

fixed query for dashboard.ts

Vaqint committed yesterday

8ec2ccc

<>

add billing table to db

bt919 committed yesterday

6093f8a

<>

add billing router, and POST endpoint

bt919 committed yesterday

c7c60c8

<>

tweak constraint on registration table

bt919 committed yesterday

8b79455

<>

Commits on Mar 19, 2024

add GET /billing route

bt919 committed 7 hours ago

d63b91f

<>

add status field to billing table

bt919 committed 7 hours ago

e899ce8

<>

add PATCH /billing route

bt919 committed 6 hours ago

f72429a

<>

merge main into db-changes














bt919 committed 6 hours ago ✓

5866edc

<>

## Use of feature branches

User stories and new features are developed on separate branches from “main” in order to maintain code quality and integrity of the main branch. Upon finishing, a merge request is made and the branch is reviewed. If no problems are found, it can be merged.

<input type="checkbox"/>	 <b>Updated Navbar</b> ✓		 1
	#51 by Irisvella was merged 2 weeks ago • Approved		
<input type="checkbox"/>	 <b>Landing Management On-clicks Update (Dashboard and Listing)</b> ✓		 1
	#48 by Zeebruh326 was merged 3 weeks ago • Approved		
<input type="checkbox"/>	 <b>Nav column (haven't update on-clicks)</b> ✓		 1
	#47 by Zeebruh326 was merged 3 weeks ago • Approved		
<input type="checkbox"/>	 <b>Enable testing and build for web-app portion + Jest set up</b> ✗		
	#46 by Irisvella was closed 5 days ago • Review required		
<input type="checkbox"/>	 <b>Employees roles tables</b> ✓ <span>sprint 2</span> <span>user story</span>		
	#45 by sarahabellard was closed 3 weeks ago • Review required		
<input type="checkbox"/>	 <b>Dashboard</b> ✓		 3
	#44 by Ashx11 was merged 3 weeks ago • Approved		
<input type="checkbox"/>	 <b>Create edit listing</b> ✓		 1
	#39 by Fatoumatabintabarry was merged 3 weeks ago • Approved		
<input type="checkbox"/>	 <b>Front-end , EditListingPage</b> ✓		 2
	#38 by Fatoumatabintabarry was closed 3 weeks ago • Changes requested		
<input type="checkbox"/>	 <b>Add files via upload</b> ✓		 1
	#37 by Ashx11 was merged 2 weeks ago • Approved		
<input type="checkbox"/>	 <b>Landing management</b>		 1
	#36 by Zeebruh326 was merged 3 weeks ago • Approved		
<input type="checkbox"/>	 <b>Add files via upload</b>		 1
	#35 by Ashx11 was closed 3 weeks ago • Review required		
<input type="checkbox"/>	 <b>Integration testing for authentication function</b> ✓		 1
	#34 by Irisvella was merged 3 weeks ago • Approved		





## Atomic commits

Team members are encouraged to commit often and to keep commits focused. This is to help other members stay aware of progress of their work. This mainly helps keep the team up to date and organized, but also helps avoid multiple people working on the same feature and only realizing much too late when they make a larger commit. On a broader level however, this practice makes code review, finding bugs, and reverting changes if need be extremely efficient.

Commits on Mar 17, 2024	
add db schemas for operating_fee bt919 committed 2 days ago	2a3fb0f <>
Commits on Mar 18, 2024	
fixed query for dashboard.ts Vaqint committed yesterday	8ec2ccc <>
add billing table to db bt919 committed yesterday	6093f8a <>
add billing router, and POST endpoint bt919 committed yesterday	c7c60c8 <>
tweak constraint on registration table bt919 committed yesterday	8b79455 <>
Commits on Mar 19, 2024	
add GET /billing route bt919 committed 1 hour ago	d63b91f <>
add status field to billing table bt919 committed 1 hour ago	e899ce8 <>
add PATCH /billing route bt919 committed 30 minutes ago	f72429a <>
merge main into db-changes bt919 committed 9 minutes ago ✓	5866edc <>

## Bug reporting

In the issues tab of our github repository, bugs are reported and labeled to keep track. When they are addressed, the issue can be marked as closed. All necessary information about the bug can be found in the “bug” issue.

<input type="checkbox"/>	<input checked="" type="radio"/>	#63 Build Failure bug Request2Management (merge #62) <span>bug</span>	
#63 opened 2 days ago by Fatoumata Barry			
<input type="checkbox"/>	<input checked="" type="radio"/>	#57 Create (add) new property listing bug. <span>bug</span>	
#57 opened 2 weeks ago by Fatoumata Barry			
<input type="checkbox"/>	<input checked="" type="radio"/>	#54 As a condo management company, I want to be able to edit properties under our management to be able to view and manage the data related to them (4 USP) <span>sprint 3</span> <span>user story</span>	
#54 opened 2 weeks ago by Fatoumata Barry			
<input type="checkbox"/>	<input checked="" type="radio"/>	Carousel Bug - displaying vertical stack <span>bug</span>	
#50 opened 2 weeks ago by Vaqint			
<input type="checkbox"/>	<input checked="" type="radio"/>	No routes to escape login/signup/profile loop <span>minor bug</span>	
#31 by Irisvella was closed last month			



## Use of issue labels for tracking and filtering

Labels such as “user story”, “minor bug”, “sprint 1”, etc... are used to identify and categorize issues.

☐

Author ▾

Label ▾

Projects ▾

Milestones ▾

Assignee ▾

Sort ▾

☐

☒

**As a user, I want to be able to quickly navigate between my profile and dashboard using the navbar (2USP)** user story

#49 by Irisvella was closed 2 weeks ago

☐

☒

**No routes to escape login/signup/profile loop** minor bug

#31 by Irisvella was closed last month

1

☐

☒

**#27 As a condo management company, I would like to be able to have a dashboard that displays all the functionalities available to me , so that i can easily navigate through them and have a general overview . (3 USP)** sprint 2

#27 by Fatoumatabintabarry was closed 2 weeks ago

1

☐

☒

**As a public user, I need to enter a registration key from my condo management company to register as a rental user, gaining access to relevant functionalities.** user story

#17 by Fatoumatabintabarry was closed on Feb 3

☐

☒

**#12 As a user of the condo management system, I would like to sign in to the condo management system to be able to use the features related to my user type (USP 4)** sprint 1 user story

#12 by sarahabellard was closed 3 weeks ago

1

☐

☒

**#7 As a condo management company, I want to be able to create properties under our management to be able to view and manage the data related to them (4 USP)** sprint 2 user story

#7 by sarahabellard was closed 2 weeks ago

1

☐

☒


**#2 As a public user, I want to upload a profile picture during account creation, so my profile feels personalized. (3 USP)** sprint 2 user story

#2 by sarahabellard was closed 3 weeks ago

1

## Links between commits and bug reports/features

☒

 Fatoumatabintabarry mentioned this pull request 34 minutes ago

**#54 As a condo management company, I want to be able to edit properties under our management to be able to view and manage the data related to them (4 USP) #54**

Closed