

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;

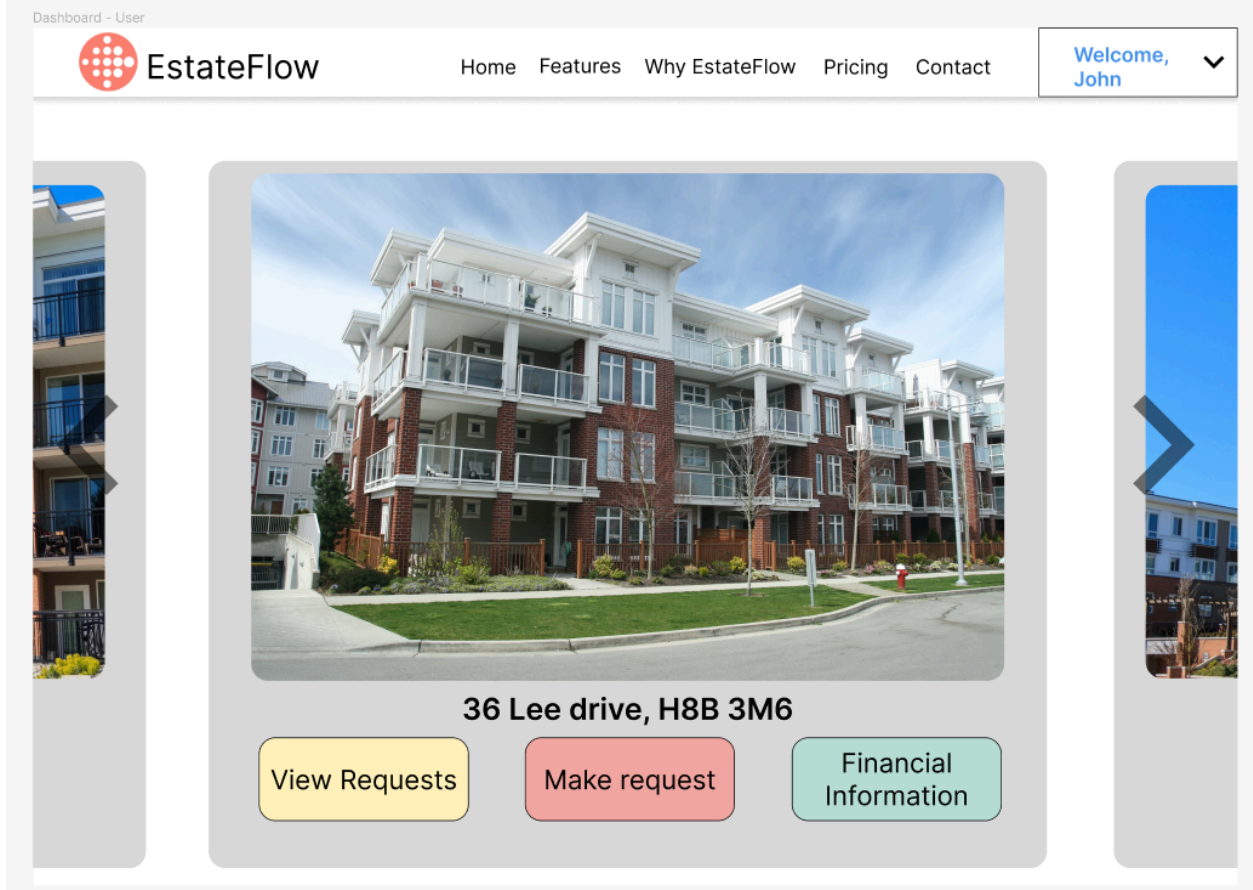
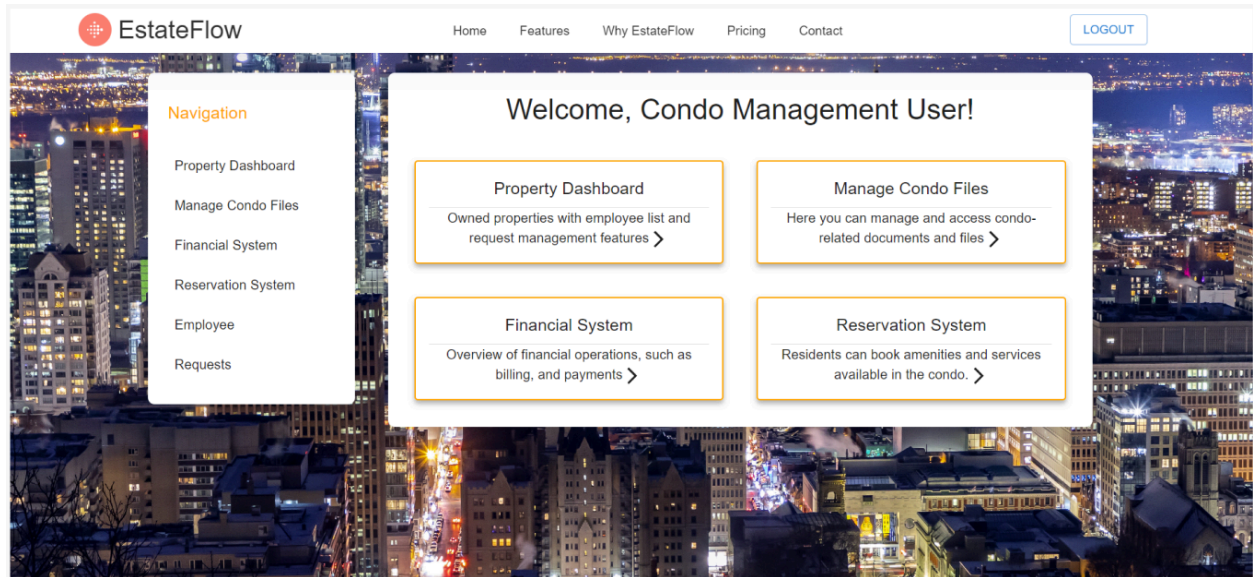
            if (role === "company") {
              const body = req.body; //constant

              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

In our design, we wanted to create a modern, minimalist and simple design with an extremely simple UI so that people with any technological understanding level can use this as an everyday app. We focused on clarity, simplicity, ease of use, with a focus on only showing users what is necessary for their user type. By keeping the UI uncluttered and focused, UX should be enhanced and speed up the learning process.

This simplicity also lends itself well to improving technological needs such as adaptability since simple designs are much easier to make adaptive to any screen size or platform. It also benefits performance since fewer elements are being processed and rendered, leading to faster loading times. This is not as important but can definitely be a plus when considering slower, older devices as well as networks.



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}
        />
        <TextField
          type="text"
          name="role"
          value={employeeData.role}
          onChange={handleChange}
        />
        <Button type="button" onClick={handleSubmit}>Add Employee</Button>
      </div>
    </div>
  );
};
```

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.

The screenshot shows a GitHub pull request interface for a pull request titled "db changes and billing route #69". At the top, it indicates that the pull request is "Merged" and that "sarahabellard merged 9 commits into main from db-changes" 4 hours ago. Below this, there are tabs for "Conversation" (1), "Commits" (9), "Checks" (2), and "Files changed" (4). A status bar shows "+392 -68" with a green progress indicator. The main content area displays a list of commits grouped by date. The first group is "Commits on Mar 17, 2024" with one commit: "add db schemas for operating_fee" by user "bt919" committed 2 days ago, with commit hash "2a3fb0f". The second group is "Commits on Mar 18, 2024" with four commits: "fixed query for dashboard.ts" by "Vaqint" committed yesterday (hash "8ec2ccc"), "add billing table to db" by "bt919" committed yesterday (hash "6093f8a"), "add billing router, and POST endpoint" by "bt919" committed yesterday (hash "c7c60c8"), and "tweak constraint on registration table" by "bt919" committed yesterday (hash "8b79455"). The third group is "Commits on Mar 19, 2024" with four commits: "add GET /billing route" by "bt919" committed 7 hours ago (hash "d63b91f"), "add status field to billing table" by "bt919" committed 7 hours ago (hash "e899ce8"), "add PATCH /billing route" by "bt919" committed 6 hours ago (hash "f72429a"), and "merge main into db-changes" by "bt919" committed 6 hours ago (hash "5866edc") with a green checkmark icon.

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.







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) bug	
		#63 opened 2 days ago by Fatoumata Bint Barry	
<input type="checkbox"/>	<input checked="" type="radio"/>	#57 Create (add) new property listing bug. bug	
		#57 opened 2 weeks ago by Fatoumata Bint 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) sprint 3 user story	
		#54 opened 2 weeks ago by Fatoumata Bint Barry	
<input type="checkbox"/>	<input checked="" type="radio"/>	Carousel Bug - displaying vertical stack bug	
		#50 opened 2 weeks ago by Vaqint	
<input type="checkbox"/>	<input checked="" type="radio"/>	No routes to escape login/signup/profile loop minor bug	
		#31 by Irisvella was closed last month	1

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