

# **Potato Properties Web Application**

The Potatoes - Group 03

Jade Lowe, Dakota Chanthakoummane, Aryan Kharva, Quintin Obey, Connor Smith, Edward  
Tobiasson

Professor Krasniqi

ITIS – 3300 – 092

October 29, 2024

## **Table of Contents**

1 Requirements	3
1.1 Project Scope	3
1.2 Phase Progress	3
1.2 Phase Requirements	3
2 Diagrams	5
2.1 Class Diagram	5
2.2 Sequence Diagram	5
2.3 Use Case Diagram	6
3 User Manual	6
3.1 Installation	6
4 Instructions	6
4.1 Compilation	6
4.2 Test Cases	7
5 Reflection	7
6 Member Contributions	7

## 1 Requirements

### 1.1 Project Scope

The scope of the Potato Properties web application encompasses user authentication for client user types. The tenant user should be able to view apartment complex information, view current and upcoming vacancies, and apply for an apartment. The current tenant user should be able to view their lease documentation, make payments, and request unit maintenance.

### 1.2 Phase Progress (Phase II)

This phase took a massive pivot. Previously, we were utilizing python and Flask (a python-based web micro framework) to create a web application. The problem was that only one of our members knew how to use this language/framework. So we decided to transition our project to be an Express, Node.js, MongoDB project. With this new tech stack, more members were able to avoid the technical overhead of learning a new language/framework, so we could do more in a shorter amount of time. Since we have more members familiar with the language/framework, we were able to bring back the management vs. client side operations and add more to our application.

### 1.2 Phase Requirements

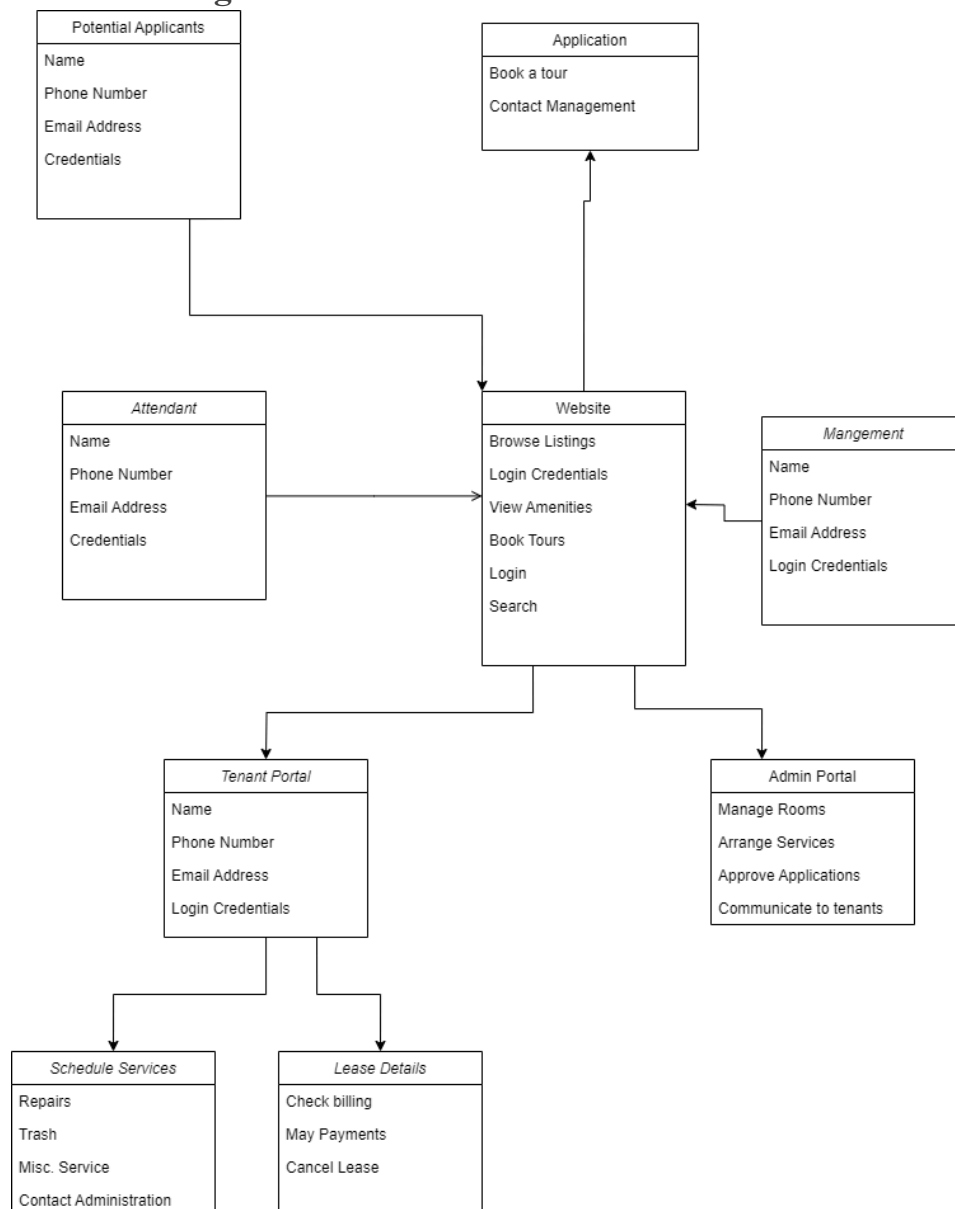
Feature	Phase	Task	Description	Technologies
Tenant UI & Pages	1	Create pages for the tenant facing side of the application.	Base UI for starting tenant pages (Subject to change as new functionality is introduced)	HTML/CSS, JS, EJS
Create User Account	1	Allow user to create an account	The user can create an account.	Express Node.js, Mongoose MongoDB
User Authentication	1	Implement user login	Implement login functionality for tenants. Tenants should be redirected to the dashboard page upon successful login.	Express Node.js, Mongoose
Authentication Validation	1	Create flash messaging for users that are logging in and/or creating an account.	Users should see a flash message for invalid input on registration and login as well as a flash message for successful account creation.	Express Node.js, Mongoose
Tenant Maintenance	2	Implement tenant maintenance requests	Implement functionality for tenants to submit and track maintenance requests	Express Node.js, Mongoose MongoDB

Payment	2	Implement payment processing system	Implement functionality for simulated payment processing	Express Node.js, Mongoose MongoDB
Payment Tracking	2	Implement payment tracking	Allow tenants to track payments	Express Node.js, Mongoose MongoDB
Application Submission	2	Implement application submission system	Implement functionality for simulated application submission from potential tenants	Express Node.js, Mongoose MongoDB
Application Tracking	2	Implement application tracking	Allow potential tenants to track application	Express Node.js, Mongoose MongoDB
Contact admin via the contact page	2	Implement a function to show a message that the user wants to send admin via contact us page.	Users should be able to send a message to admin via the contact us page. A function can be implemented to show this message, simulating an email.	Express Node.js, Mongoose MongoDB
Document Submission	2	Implement document upload and storage system	Allow both tenant types to upload relevant documents	Express Node.js, Mongoose MongoDB
Document Viewing	2	Implement document viewing	Allow both tenant types to view relevant documents	Express Node.js, Mongoose MongoDB
Lease Information	2	Display lease information	Allow current tenants to view their lease information	Express Node.js, Mongoose MongoDB
Improve UI	3	Update UI to look better	All UI Pages receive updated styling to improve flow of the pages.	Express Node.js, Mongoose MongoDB
Security Update	3	Implement user data encryption	Encrypt user data for enhanced security	Bcryptjs
Optimize performance	3	Optimize website performance	Optimize web page loading times and page responsiveness	Express Node.js, Mongoose MongoDB

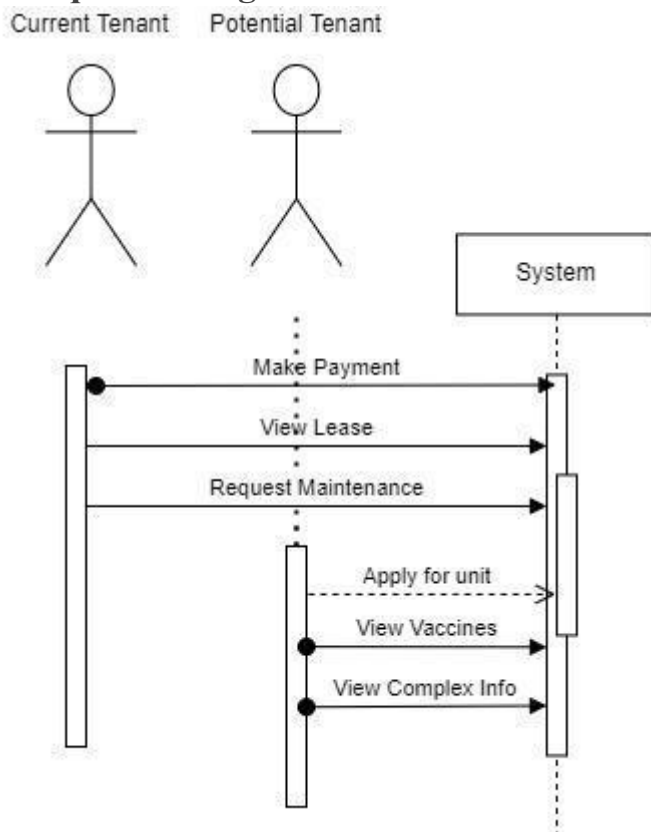
Write Test Cases	3	Test different endpoints for user, property, and maintenance requests	Write test cases to ensure that each endpoint, post and get endpoints, are working as intended using the Jest framework and Supertest library.	Node.js, Jest, Supertest
------------------	---	---	--	--------------------------

## 2 Diagrams

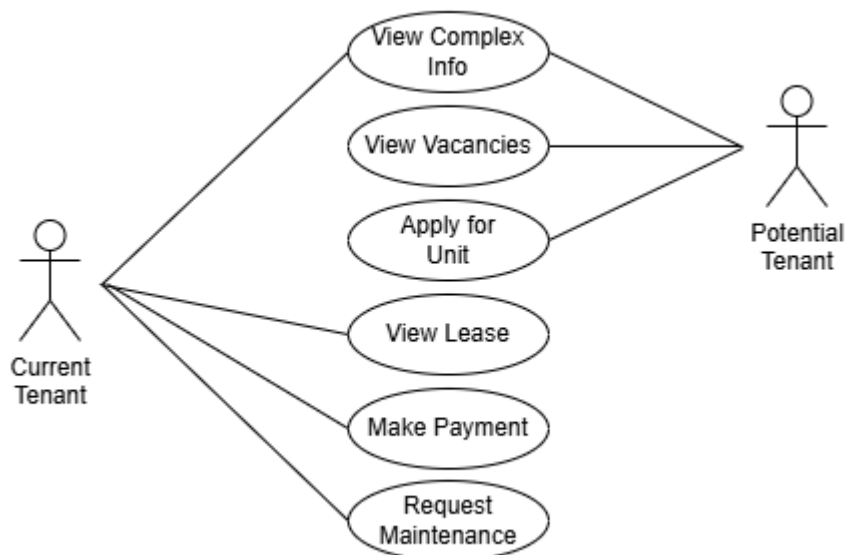
### 2.1 Class Diagram



## 2.2 Sequence Diagram



## 2.3 Use Case Diagram



## **3 User Manual**

### **3.1 Installation**

Clone this repository to your local machine using git clone

<https://github.com/jlowe093/ITIS3300-GroupProject.git>

## **4 Instructions**

### **4.1 Compilation**

Change the directory to the project folder (\src).

Open a terminal in this project folder and run `npm i` to install dependencies.

Now to start the web server, run `node app`.

Go to <http://localhost:3000> to see the running application.

Note: You must have git and Node.js installed to run these commands.

### **4.2 Test Cases**

#### **Navigation Testing**

To test this project, we ran the project and tested many functionalities that were working as intended. For the **user interface**, we tested that all buttons/links are responsive as expected such as the login and contact us. This allowed for fluid navigation through the website letting the user find the information they wanted to find.

Furthermore, we tested the **login functionality**, by entering valid and invalid credentials to see if the form was able to handle these instances properly. If the wrong credentials were input then errors would be displayed and on the flipside the correct allowed login.

We also ensured that the **property listings** were populated with the correct images, prices, and features. This is vital as customers who visit the website can confidently make choices based on accurate information.

#### **Unit Testing**

Furthermore, we utilized the **Jest Framework** and the library **supertest** to create unit tests for our user routes thus far. This framework allowed us to test GET/POST routes to ensure that 200/302 status codes were achieved on successful cases, and 404/400 errors were shown on error cases.

Below is our test cases for userController after running:

```
GET /users/applications/673a20f6da5bc40dc5ac5347 404 1563 - 4.708 ms
PASS __tests__/controllers/userController.test.js
  ✓ Get /users returns 404 NF (811 ms)
  ✓ Get /users/new returns 200 OK (55 ms)
  ✓ Get /users/contact returns 200 OK (49 ms)
  ✓ Get /users/login returns 200 OK (65 ms)
  ✓ POST /users/login returns 302 redirect on Successful Login (179 ms)
  ✓ POST /users/login returns 400 client error on Unsuccessful Login (96 ms)
  ✓ GET /users/dashboard with LoggedIn User gives 200 OK on dashboard page (391 ms)
  ✓ GET /users/dashboard with a Guest User gives 302 redirect to login page (198 ms)
  ✓ GET /users/application/:id with an invalid property id gives a 404 error (312 ms)
  ✓ GET /users/application/:id with a valid property id, but on a Non-Management user gives a 403 error (272 ms)

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        3.491 s, estimated 4 s
Ran all test suites matching /__tests__\controllers\userController.test.js/i.
```

Note: We still need to create tests for the other two controllers and middleware.

## 5 Reflection

In this phase, we successfully implemented key features such as account creation and login functionality using Express, integrated with a cloud-based MongoDB Atlas database. Additionally, we utilized Nodemon for efficient development and established user types for potential tenants, maintenance staff, and management. Potential tenants can now submit applications seamlessly through the platform. While significant progress has been made, areas requiring improvement include configuring the maintenance dashboard, enabling tenants to submit maintenance requests, and cleaning up the interfaces throughout the site. These updates will enhance the overall usability and functionality of the application as it nears completion.

## 6 Member Contributions

<u>Member Name</u>	<u>Contribution Description</u>	<u>Overall % of Contribution</u>	<u>Note</u> (if applicable)
Jade Lowe	Report formatting, use case diagram, implement various aspects of code to allow for a more in depth front end and allow for adding properties and users, team collaboration	16.66	



Dakota Chanthakoummane	Report contribution, team collaboration	16.66	
Aryan Kharva	frontend for the login, signup, further implemented more front end aspects, and dashboard pages, team collaboration	16.66	
Quintin Obey	Report contribution, sequence diagram, team collaboration	16.66	
Connor Smith	Slotted html, created database, implement various code to add to the dynamic aspect of the project, report contribution, team collaboration	16.66	
Edward Tobiasson	Report contribution, team collaboration	16.66	