San Francisco State University

SW Engineering CSC 648/848

Milestone 3

November 06, 2024

Section 04 — Team 02

Members

Katy Lam	**Team Lead**
Arjun Singh Gill	**Back-end**
Matthew Aaron Weesner	**Back-end**
Niko Galedo	**Front-end**
Kevin Lam	**Front-end**
Kullathon “Mos” Sitthisarnwattanachai	**Git Master**
Arizza Cristobal	**Scrum Master**

Revision History

Version	Date	Summary
1.0	2024-11-07	First version submitted for instructor approval
0.2	2024-11-06	Update information after Demo
0.1	2024-10-23	Initial draft proposed to Team Lead

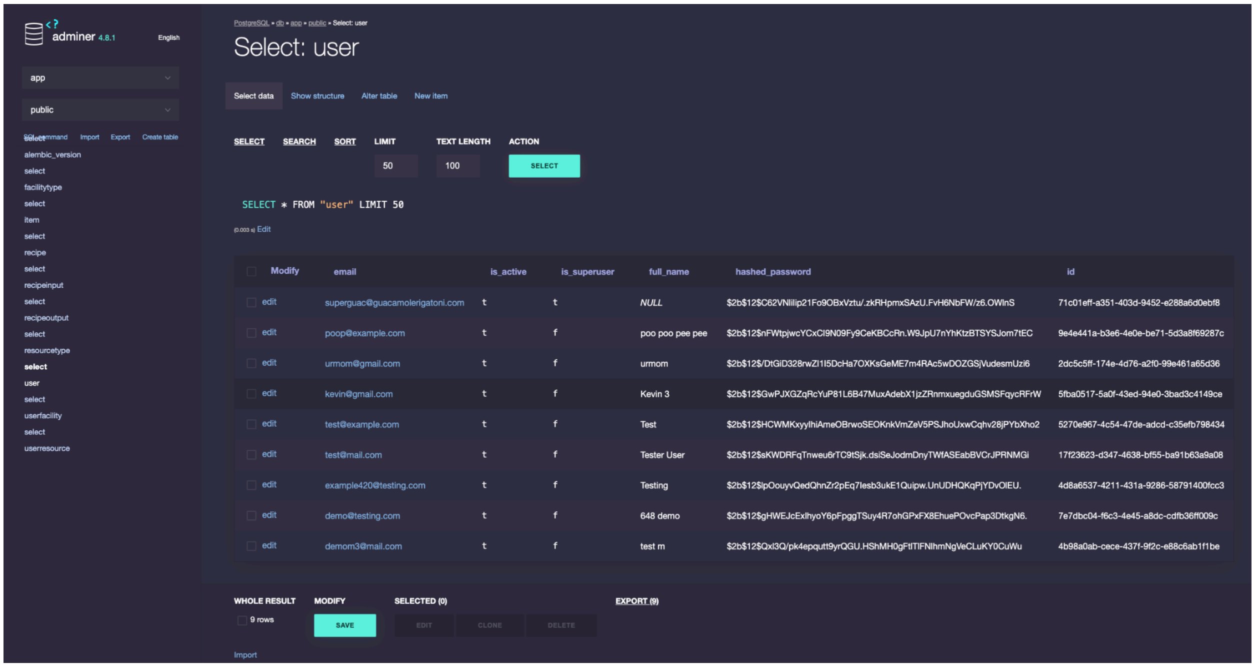
UI and functionality feedback (P1 functions only)

Instructor's comments on UI/functionality for your demo (should be during the class of M3 demo)

- Is Backend & Frontend Connected
- What are the differences between Minecraft and Click & Mortar functionalities.
    - Clarifications on Minecraft images vs logic behind code
- What are our P2 Functionalities

Your Plan for the comments

- The backend and frontend are already connected using Adminer.
- Each user account includes a unique ID and a securely hashed password.



LIMIT: 50

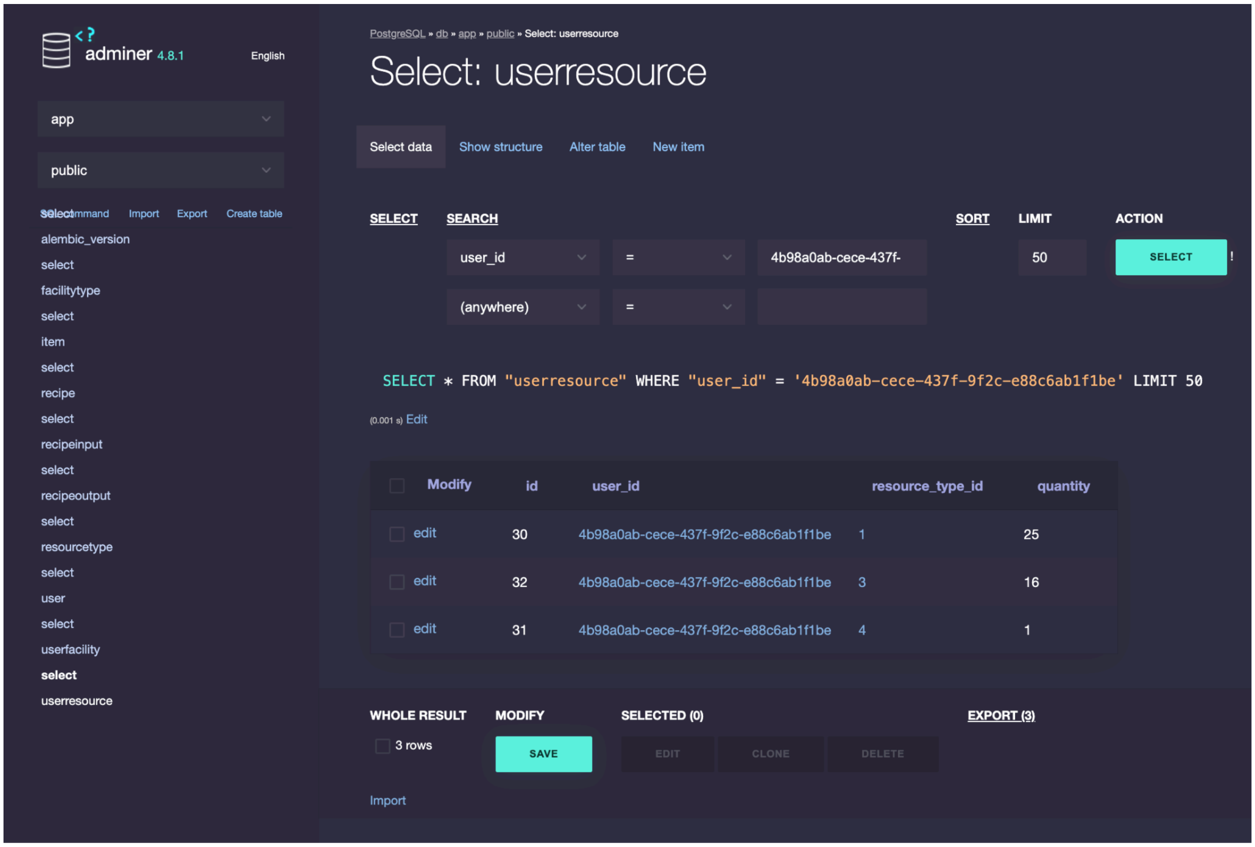
TEXT LENGTH: 100

```
SELECT * FROM "user" LIMIT 50
```

□ Modify	email	is_active	is_superuser	full_name	hashed_password	id
□ edit	superguac@guacamolerigatoni.com	t	t	*NULL*	$2b$12$C62VNlilip21Fo9OBxVztu/.zkRHpmxSAzU.FvH6NbFW/z6.OWlnS	71c01eff-a351-403d-9452-e288a6d0ebf8
□ edit	poop@example.com	t	f	poo poo pee pee	$2b$12$nFWtpjwcYCxCl9N09Fy9CeKBCcRn.W9JpU7nYhKtzBTSYSJom7tEC	9e4e441a-b3e6-4e0e-be71-5d3a8f69287c
□ edit	urmom@gmail.com	t	f	urmom	$2b$12$/DtGiD328rwZl1I5DcHa7OXKsGeME7m4RAc5wDOZGSjVudesmUzi6	2dc5c5ff-174e-4d76-a2f0-99e461a65d36
□ edit	kevin@gmail.com	t	f	Kevin 3	$2b$12$GwPJXGZqRcYuP81L6B47MuxAdebX1jzZRnmxuegduGSMSFqycRFrW	5fba0517-5a0f-43ed-94e0-3bad3c4149ce
□ edit	test@example.com	t	f	Test	$2b$12$HCWMKxyylhiAmeOBrwoSEOKnkVmZeV5PSJhoUxwCqhv28jPYbXho2	5270e967-4c54-47de-adcd-c35efb798434
□ edit	test@mail.com	t	f	Tester User	$2b$12$sKWDRFqTnweu6rTC9tSjk.dsiSeJodmDnyTWfASEabBVCrJPRNMGi	17f23623-d347-4638-bf55-ba91b63a9a08
□ edit	example420@testing.com	t	f	Testing	$2b$12$lpOouyvQedQhnZr2pEq7lesb3ukE1Quipw.UnUDHQKqPjYDvOlEU.	4d8a6537-4211-431a-9286-58791400fcc3
□ edit	demo@testing.com	t	f	648 demo	$2b$12$gHWEJcExlhyoY6pFpggTSuy4R7ohGPxFX8EhuePOvcPap3DtkgN6.	7e7dbc04-f6c3-4e45-a8dc-cdfb36ff009c
□ edit	demom3@mail.com	t	f	test m	$2b$12$Qxl3Q/pk4epqutt9yrQGU.HShMH0gFtlTlFNlhmNgVeCLuKY0CuWu	4b98a0ab-cece-437f-9f2c-e88c6ab1f1be

□ 9 rows

- Using a unique user ID to retrieve the resources associated with each user's account.



Select: userresource

SEARCH: user_id = 4b98a0ab-cece-437f-
(anywhere) =

LIMIT: 50

```
SELECT * FROM "userresource" WHERE "user_id" = '4b98a0ab-cece-437f-9f2c-e88c6ab1f1be' LIMIT 50
```

(0.001 s) Edit

□	Modify	id	user_id	resource_type_id	quantity
□	edit	30	4b98a0ab-cece-437f-9f2c-e88c6ab1f1be	1	25
□	edit	32	4b98a0ab-cece-437f-9f2c-e88c6ab1f1be	3	16
□	edit	31	4b98a0ab-cece-437f-9f2c-e88c6ab1f1be	4	1

□ 3 rows

- The game logic in Minecraft is entirely different from the logic used in Click & Mortar. For our demo, we are temporarily using Minecraft images, but in Phase 2, we will implement our own graphics. Unlike Minecraft, Click & Mortar includes unique features such as clicking and automation, which are core mechanics in our game.

P2 Functions (Future Implementation)

Priority	Functional Requirement	Description	Progress
2	Inventory Management	Users have the ability to manage items and resources in their inventory.	Not Started
2	UI/UX Graphics	Users can see our chosen/drawn graphics and user interface elements.	In Progress
2	Progression	Users are able to check on In-game progression mechanics, such as leveling or unlocking	Not Started

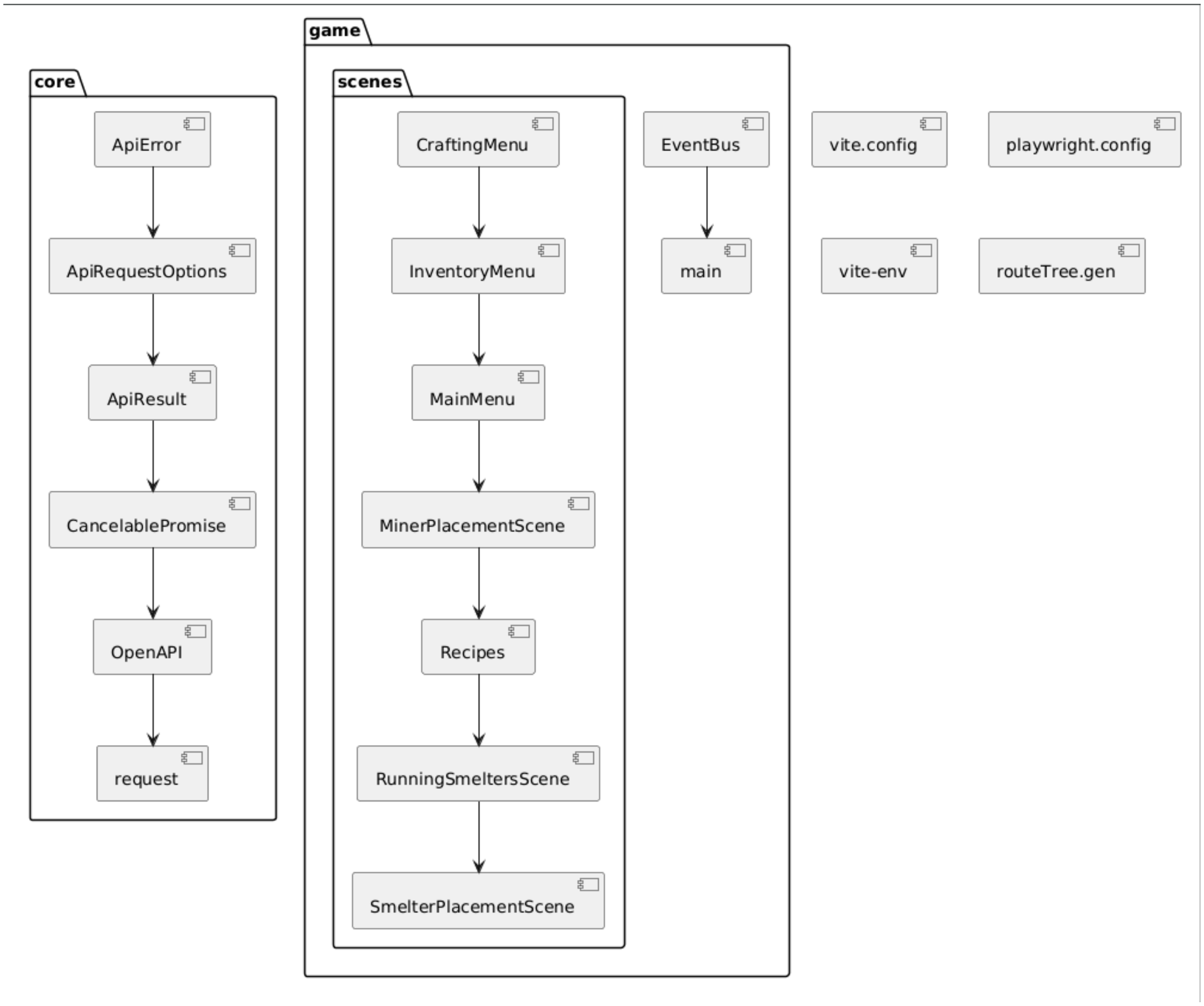
		features.	
2	Leaderboard	Users are able to access leaderboard rankings based on resource values and points.	Not Started
2	Chatbot Assistant	Users have access to a Game assistant that provides tips, formulas, recipes, and inputs.	In Progress
2	Achievement System	Users are able to Track achievements and milestones within the game.	Not Started

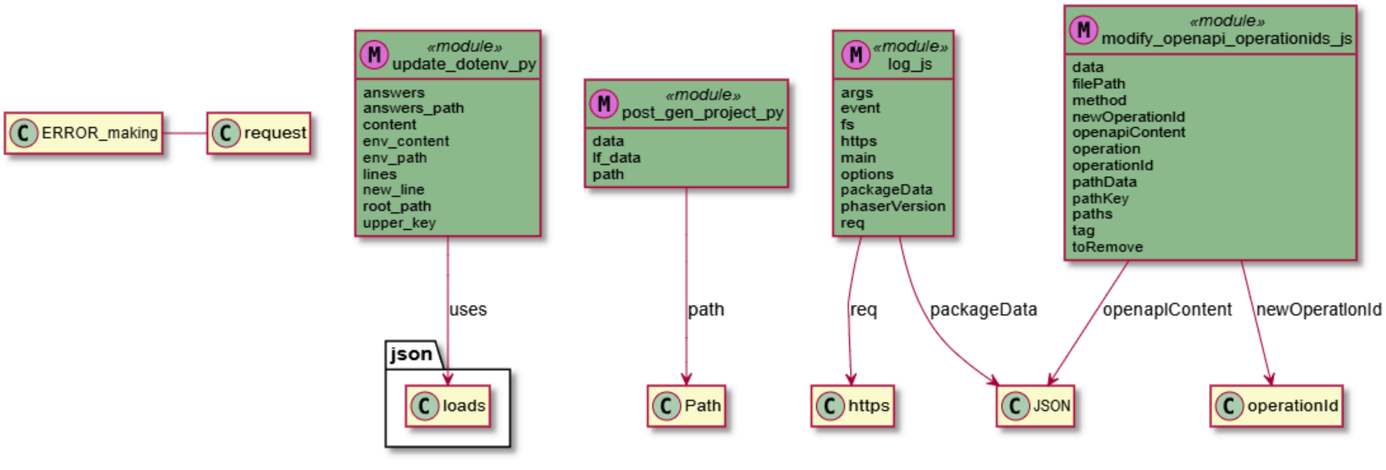
List of P1 features committed for delivery

Priority	Functional Requirement	Description	Progress
1	Login, Logout	Users can do basic user authentication functionalities such as logging in and logging out	Completed ▾
1	Manually Mine Resource	Users can manually mine resources by clicking or selecting options.	Completed ▾
1	Manually Craft	Users can craft items manually by selecting recipes and resources.	Completed ▾
1	Automated Mine Resource	Users can have an automated collection of resources using in-game miners.	Completed ▾
1	Automated Craft: Crafter	Users can automatically craft with a single input-to-output conversion per tick.	Completed ▾
1	Automated Craft: Assembler	Users can have automated crafting with multiple inputs to multiple outputs per tick.	Completed ▾

Architecture

Frontend:

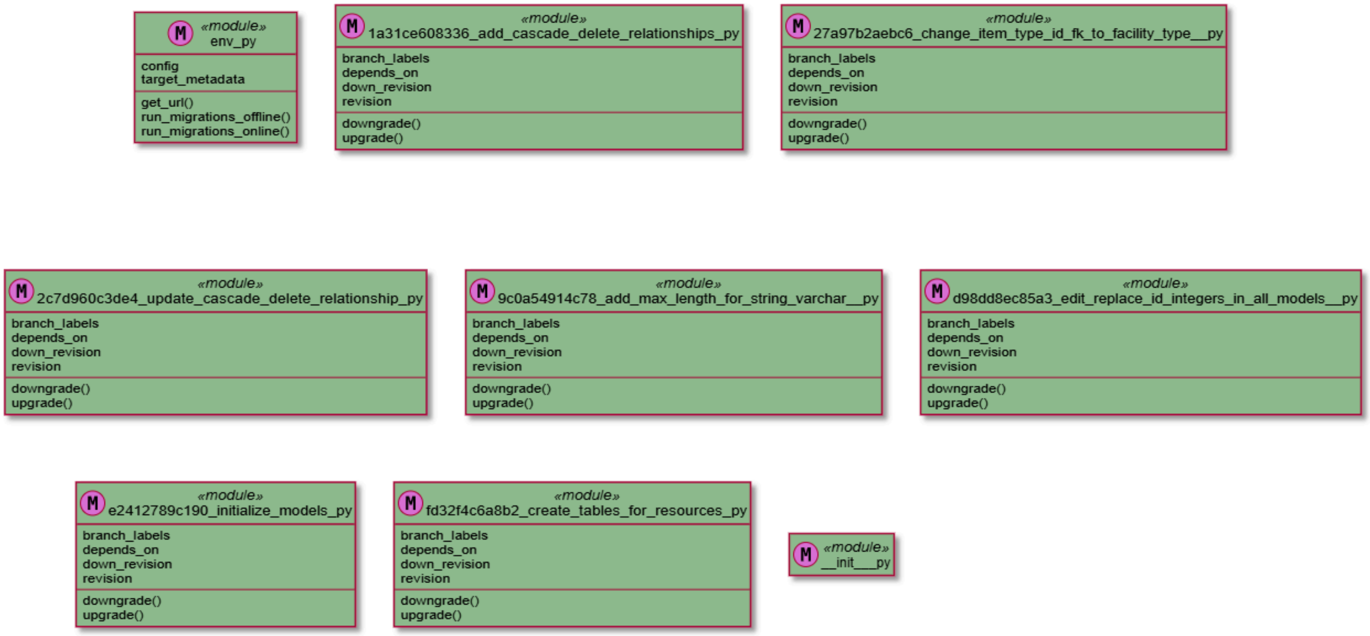


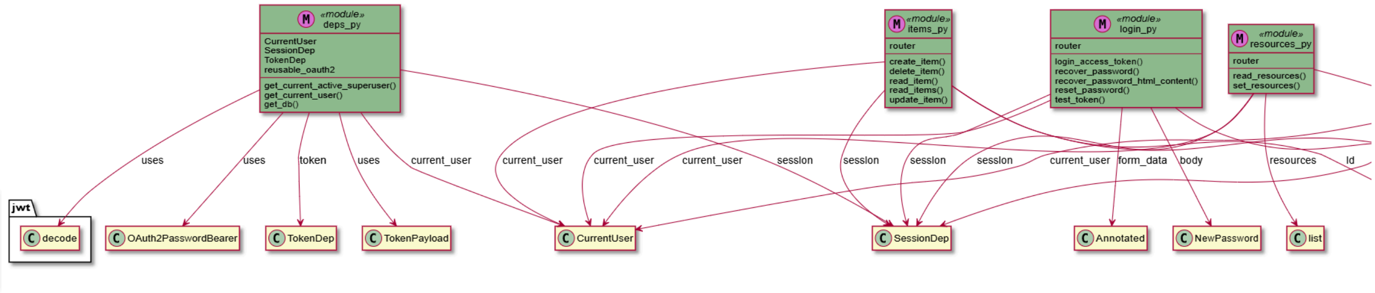


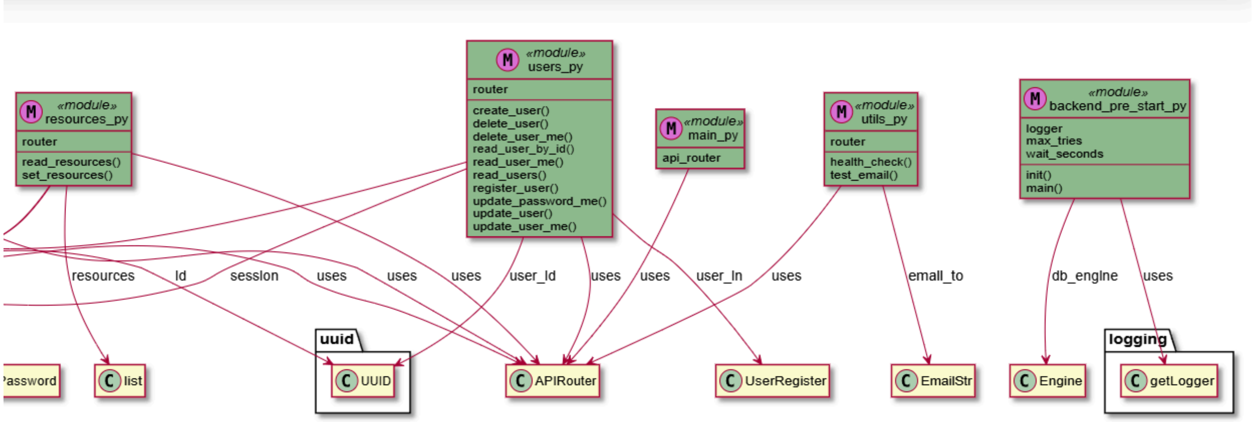
Generated by GitUml www.gituml.com

Backend

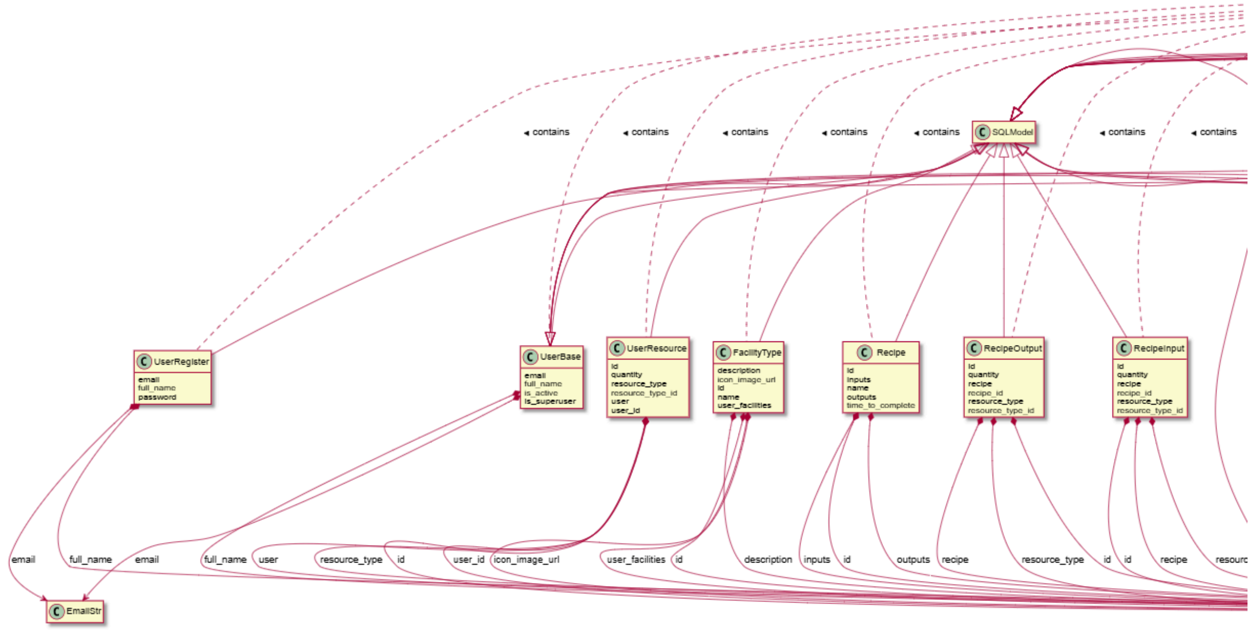
(Alembic and API Folder)

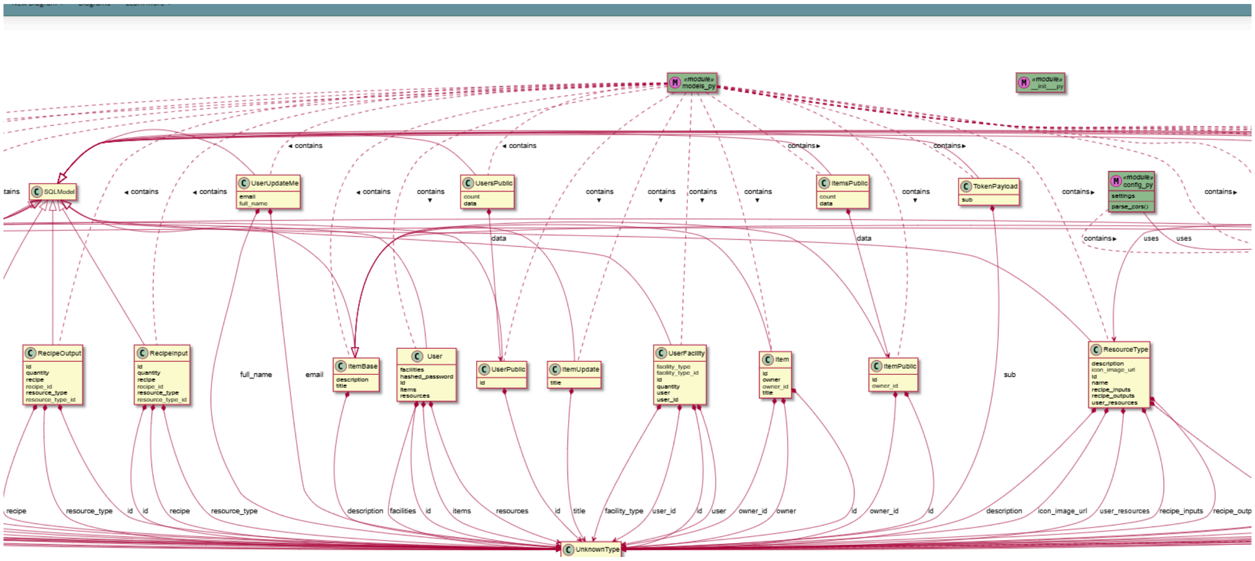


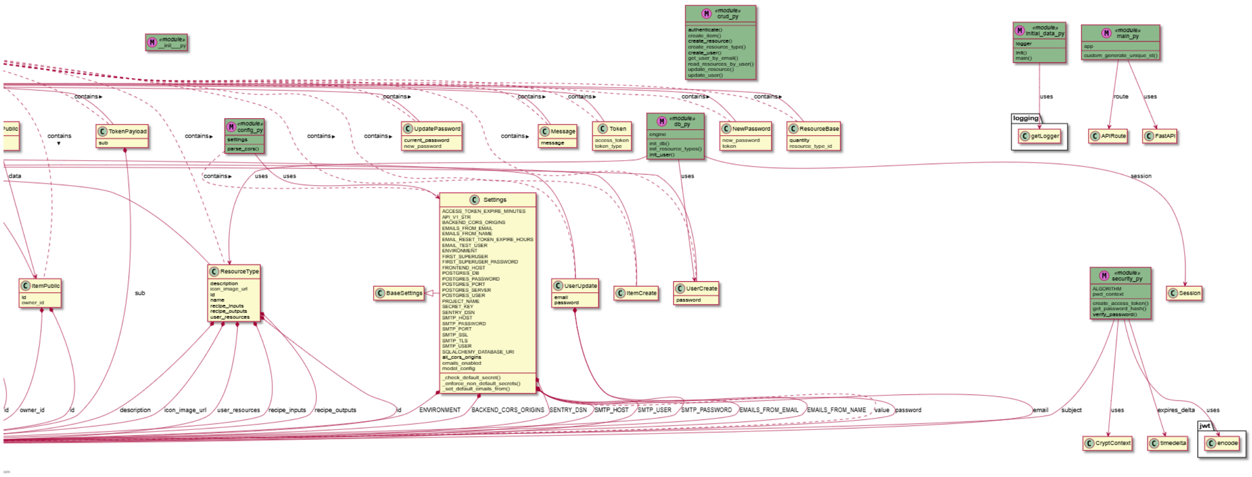

«module»
deps_py
CurrentUser
SessionDep
TokenDep
reusable_oauth2
get_current_active_superuser()
get_current_user()
get_db()
«module»
items_py
router
create_item()
delete_item()
read_item()
read_items()
update_item()
«module»
login_py
router
login_access_token()
recover_password()
recover_password_html_content()
reset_password()
test_token()
«module»
resources_py
router
read_resources()
set_resources()
uses
uses
token
uses
current_user
current_user
current_user
current_user
session
session
session
session
current_user
form_data
body
resources
id
jwt
decode
OAuth2PasswordBearer
TokenDep
TokenPayload
CurrentUser
SessionDep
Annotated
NewPassword
list


«module»
resources_py
router
read_resources()
set_resources()
«module»
users_py
router
create_user()
delete_user()
delete_user_me()
read_user_by_id()
read_user_me()
read_users()
register_user()
update_password_me()
update_user()
update_user_me()
«module»
main_py
api_router
«module»
utils_py
router
health_check()
test_email()
«module»
backend_pre_start_py
logger
max_tries
wait_seconds
init()
main()
resources
id
session
uses
uses
uses
user_id
uses
uses
user_in
uses
email_to
db_engine
uses
Password
list
uuid
UUID
APIRouter
UserRegister
EmailStr
Engine
logging
getLogger

(Core folder)


◂ contains
◂ contains
◂ contains
◂ contains
◂ contains
SQLModel
◂ contains
◂ contains
UserRegister
email
full_name
password
UserBase
email
full_name
is_active
is_superuser
UserResource
id
quantity
resource_type
resource_type_id
user
user_id
FacilityType
description
icon_image_url
id
name
user_facilities
Recipe
id
inputs
name
outputs
time_to_complete
RecipeOutput
id
quantity
recipe
recipe_id
resource_type
resource_type_id
RecipeInput
id
quantity
recipe
recipe_id
resource_type
resource_type_id
email
full_name
email
full_name
user
resource_type
id
user_id
icon_image_url
user_facilities
id
description
inputs
id
outputs
recipe
resource_type
id
id
recipe
resourc
EmailStr


«module»
models_py
«module»
__init__py
◂ contains
◂ contains
contains ▸
contains ▸
tains
SQLModel
◂ contains
◂ contains
UserUpdateMe
email
full_name
◂ contains
contains ▾
UsersPublic
count
data
contains ▾
contains ▾
contains ▾
contains ▾
ItemsPublic
count
data
contains ▾
TokenPayload
sub
contains ▸
«module»
config_py
settings
parse_cors()
contains ▸
data
data
contains ▸
uses
uses
RecipeOutput
id
quantity
recipe
recipe_id
resource_type
resource_type_id
RecipeInput
id
quantity
recipe
recipe_id
resource_type
resource_type_id
full_name
email
ItemBase
description
title
User
facilities
hashed_password
id
items
resources
UserPublic
id
ItemUpdate
title
UserFacility
facility_type
facility_type_id
id
quantity
user
user_id
Item
id
owner
owner_id
title
ItemPublic
id
owner_id
sub
ResourceType
description
icon_image_url
id
name
recipe_inputs
recipe_outputs
user_resources
recipe
resource_type
id
id
recipe
resource_type
description
facilities
id
items
resources
id
title
facility_type
user_id
id
user
owner_id
owner
id
owner_id
id
description
icon_image_url
user_resources
recipe_inputs
recipe_outp
UnknownType


«module»
__init___py
«module»
crud_py
authenticate()
create_item()
create_resource()
create_resource_type()
create_user()
get_user_by_email()
read_resources_by_user()
update_resource()
update_user()
«module»
initial_data_py
logger
init()
main()
«module»
main_py
app
custom_generate_unique_id()
contains ▸
contains ▸
contains ▸
contains ▸
contains ▸
contains ▸
uses
route
uses
Public
contains
▾
TokenPayload
sub
contains ▸
«module»
config_py
settings
parse_cors()
contains ▸
contains ▸
UpdatePassword
current_password
new_password
contains ▸
Message
message
Token
access_token
token_type
«module»
db_py
engine
init_db()
init_resource_types()
init_user()
NewPassword
new_password
token
ResourceBase
quantity
resource_type_id
logging
getLogger
APIRoute
FastAPI
data
contains ▸
uses
uses
uses
session
Settings
ACCESS_TOKEN_EXPIRE_MINUTES
API_V1_STR
BACKEND_CORS_ORIGINS
EMAILS_FROM_EMAIL
EMAILS_FROM_NAME
EMAIL_RESET_TOKEN_EXPIRE_HOURS
EMAIL_TEST_USER
ENVIRONMENT
FIRST_SUPERUSER
FIRST_SUPERUSER_PASSWORD
FRONTEND_HOST
POSTGRES_DB
POSTGRES_PASSWORD
POSTGRES_PORT
POSTGRES_SERVER
POSTGRES_USER
PROJECT_NAME
SECRET_KEY
SENTRY_DSN
SMTP_HOST
SMTP_PASSWORD
SMTP_PORT
SMTP_SSL
SMTP_TLS
SMTP_USER
SQLALCHEMY_DATABASE_URI
all_cors_origins
emails_enabled
model_config
_check_default_secret()
_enforce_non_default_secrets()
_set_default_emails_from()
ItemPublic
id
owner_id
sub
ResourceType
description
icon_image_url
id
name
recipe_inputs
recipe_outputs
user_resources
BaseSettings
UserUpdate
email
password
ItemCreate
UserCreate
password
«module»
security_py
ALGORITHM
pwd_context
create_access_token()
get_password_hash()
verify_password()
Session
id
owner_id
id
description
icon_image_url
user_resources
recipe_inputs
recipe_outputs
id
ENVIRONMENT
BACKEND_CORS_ORIGINS
SENTRY_DSN
SMTP_HOST
SMTP_USER
SMTP_PASSWORD
EMAILS_FROM_EMAIL
EMAILS_FROM_NAME
value
password
email
subject
uses
expires_delta
uses
jwt
CryptContext
timedelta
encode

Project Status

The project has made significant progress with improvements to the team's workflow and technical components. Several risks were identified and addressed to ensure a smoother development process.

Frontend Code Quality

- **Issue**: The front end needed better code quality, which could hinder future development and maintenance.
- **Resolution**: Refactoring efforts were made by delegating test cases and assigning specific tasks to team members. This helped improve code quality and ensured better coverage.

Last-Minute Work Habits

- **Issue**: Team members were treating the project as a solo effort and leaving tasks until the last minute, affecting the quality of the deliverables.
- **Resolution**: The team lead began regularly checking in with members to monitor progress. Additionally, Scrum processes were introduced to assign due dates for Jira tasks, promoting timely completion.

Presentations

- **Issue**: The approach of delegating presenting to someone was disorganized, and often decided on the day of the presentation without adequate preparation.
- **Resolution**: The presenting responsibility was delegated to the team lead to ensure consistent representation, and communication protocols such as communicating ahead of time and preparing well in advance, allowing for more improved clarifications on availability and scheduling.

UML Diagram Development

- **Issue**: There was no clear direction for UML diagrams, and there was a lack of coordination between the frontend and backend teams.
- **Resolution**: The team lead began communicating with the frontend team to gather inputs, then relayed this information to the backend team to ensure alignment and consistency in the diagrams.

The team has addressed key risks related to scheduling, communication, and technical quality. Regular check-ins, clearer delegation, and improved planning processes have been implemented to prevent these issues from recurring.

Team

Members

The list of team member names and their roles are repeated here:

Member	Role
Katy Lam	Team Lead
Arjun Singh Gill	Back-end
Matthew Aaron Weesner	Back-end
Niko Galedo	Front-end
Kevin Lam	Front-end
Kullathon “Mos” Sitthisarnwattanachai	Git Master
Arizza Cristobal	Scrum Master

M3 Checklist

The following checklist have the requirements for Milestone 3.

Item	Status
1. **REVIEW P1 Functionalities with Team** ○ During 10/23 team meeting, review and confirm P1 functionalities that need to be worked on during Sprint 5-6	DONE ▾
2. **COMPLETE PRE-DEMO P1 Functionalities** ○ Sprint 5-6 P1 functionality are completed	DONE ▾
3. **REVIEW POST-DEMO P1 Functionalites with Team** ○ After M3 demo, review P1 notes and fix any changes that need to be made	DONE ▾
4. **COMPLETE POST-DEMO P1 Functionaliites** ○ Describe	DONE ▾
5. **Resolve & Update Project Status** ○ During Sprint 5-6, SCRUM needs to note down any risks that happen and update it in M3.	DONE ▾
6. **Architecture** ○ Team Lead completes High-level sequence diagrams: for ~5~6 functional requirements by communicating with front end and back end	DONE ▾
7. **Horizontal SW Prototype** ○ P1 functions are demoed and working properly ○ It is well-organized, properly documented, and deployed on the server.	DONE ▾

- It is submitted correctly following the email process.