All in One

G3T3

Members: Benedict Chia, Joel Sng, Kasen, Leong Yew Kit, Hong Yan Jie

SMU Singapore Management University | School of Computing and Information Systems

How we conducted our scrum process

What we Did

Scrum Process

Trunk Based Development
- Each dev is assigned a team
- Each team has a branch
- ‘Backend, Frontend’
- Each team makes frequent commits to their branch
- Integrated once completed

Task Assignment
- Each user story is broken down into ‘Frontend’, ‘Backend’, ‘Integration’, ‘Testing’
- Assigned in Jira to a developer

Scrum Ceremonies

Story Point Estimation
- Planning Poker
- Alignment of developers on expected outcomes

Format
- Physical/Online
- Feedback and reflection was encouraged

Tools Used

Jira zoom

Best practices we learned/developed

Leveraging TDD
- Increasingly relied on TDD
- Ensured reliability, correctness, and stability of code
- Confidence in code performance

Logical Flow of Information
- We began each meeting with a recap of the last
- Embedded previous meeting documents
- Refreshed our memory about reflections, learnings, feedback, and opportunities

zoom AI Companion

Confluence

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Computing and Information Systems

How we estimated our product backlog items

Process

Planning Poker

- Fibonacci Sequence [1 , 2 , 3 , 5 , 8]
- Privately pick values
- Repeat and discuss until consensus

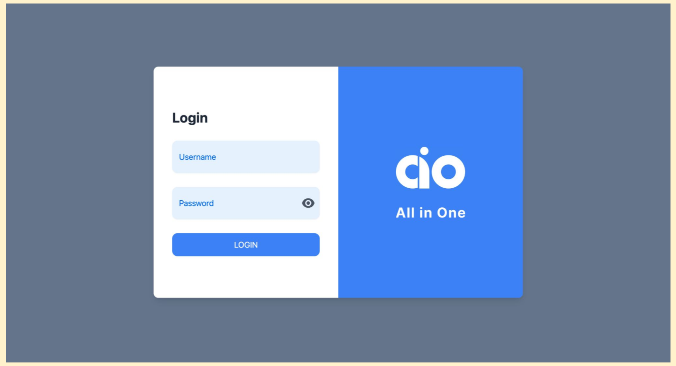
How we Changed

- First sprint we estimated points on our developer prior experience and ‘feel’
- Second and third sprint, we devised more concrete documentation to judge story points by

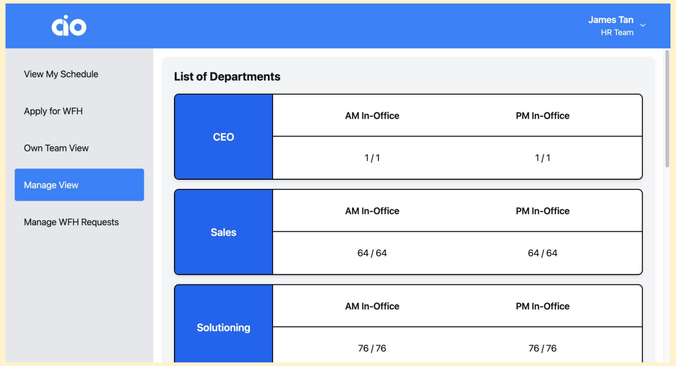
Documentation of Estimation Baseline

- In team Confluence
- Estimation baseline and process
- Code snippets that detail what constitutes **1 story point** for both frontend and backend
- Determined by consensus of our developers

Examples



1 story point



List of Departments

	AM In-Office	PM In-Office
CEO	1 / 1	1 / 1
Sales	64 / 64	64 / 64
Solutioning	76 / 76	76 / 76

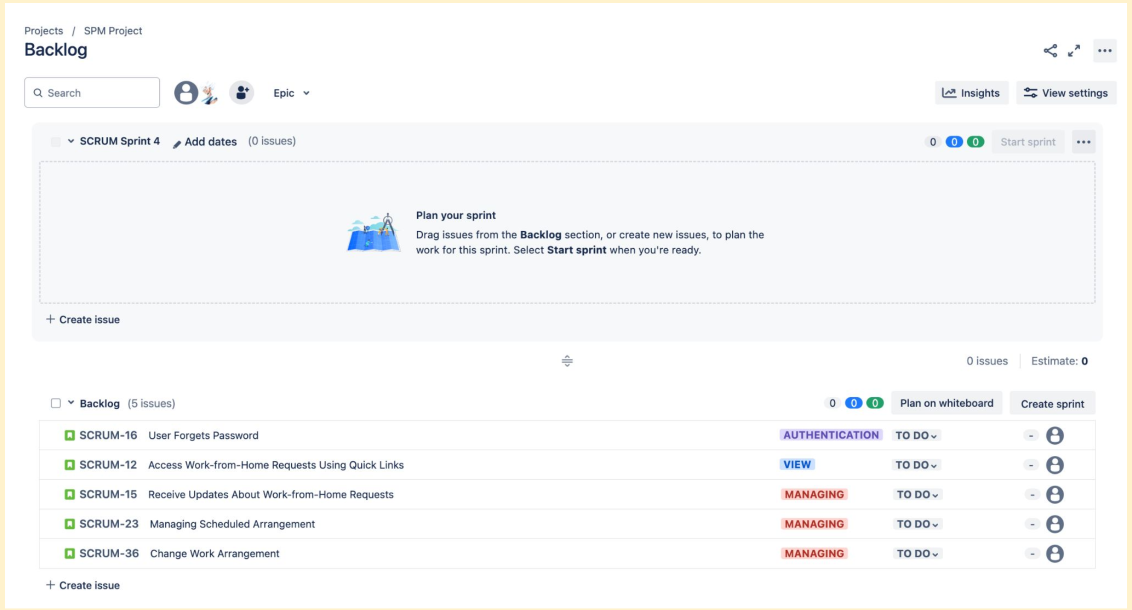
8 story points

SMU Singapore Management University | School of Computing and Information Systems

What our product backlog, user stories, & DoD look like

Product Backlog

- Managed in Jira
- 3 Epics: Authentication, View, Managing
- Every User Story belongs to an epic



Projects / SPM Project

Backlog

Search | Epic | Insights | View settings

SCRUM Sprint 4 · Add dates (0 issues) · 0 0 0 · Start sprint

Plan your sprint

Drag issues from the **Backlog** section, or create new issues, to plan the work for this sprint. Select **Start sprint** when you're ready.

+ Create issue

0 issues · Estimate: 0

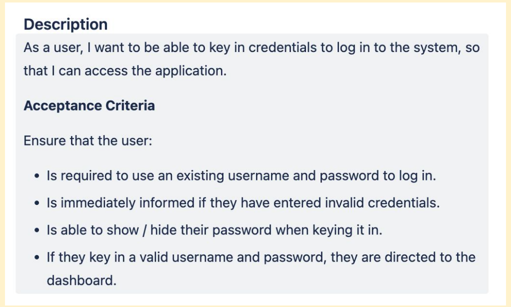
Backlog (5 issues) · 0 0 0 · Plan on whiteboard · Create sprint

SCRUM-16	User Forgets Password	AUTHENTICATION	TO DO	-
SCRUM-12	Access Work-from-Home Requests Using Quick Links	VIEW	TO DO	-
SCRUM-15	Receive Updates About Work-from-Home Requests	MANAGING	TO DO	-
SCRUM-23	Managing Scheduled Arrangement	MANAGING	TO DO	-
SCRUM-36	Change Work Arrangement	MANAGING	TO DO	-

+ Create issue

User Stories

- User story description and AC are documented in Jira



Description

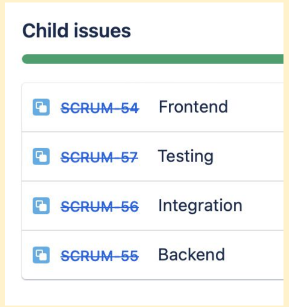
As a user, I want to be able to key in credentials to log in to the system, so that I can access the application.

Acceptance Criteria

Ensure that the user:

- Is required to use an existing username and password to log in.
- Is immediately informed if they have entered invalid credentials.
- Is able to show / hide their password when keying it in.
- If they key in a valid username and password, they are directed to the dashboard.

- Each User Story is split into 4 child issues, and each is assigned to a dev

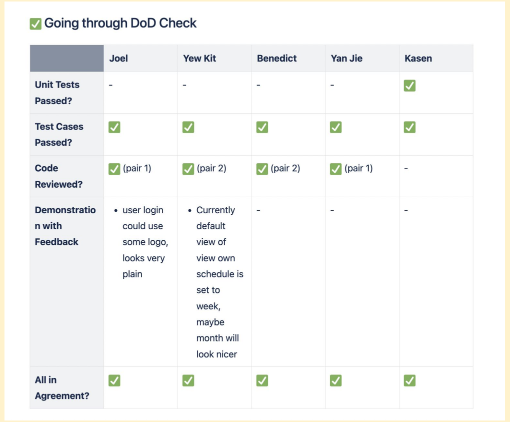


Child issues

~~SCRUM-54~~	Frontend
~~SCRUM-57~~	Testing
~~SCRUM-56~~	Integration
~~SCRUM-55~~	Backend

Definition of Done

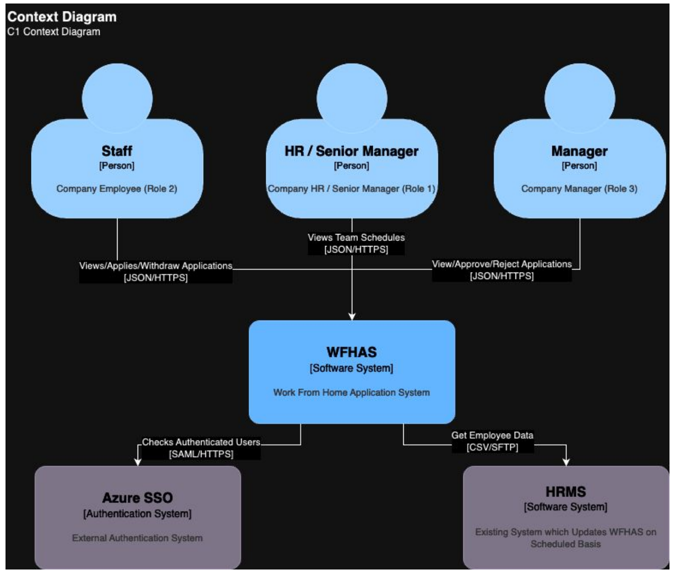
- Unit tests passed
- Code has been reviewed by peer
- Majority of team accepts the user story is done
- Test Cases Passed



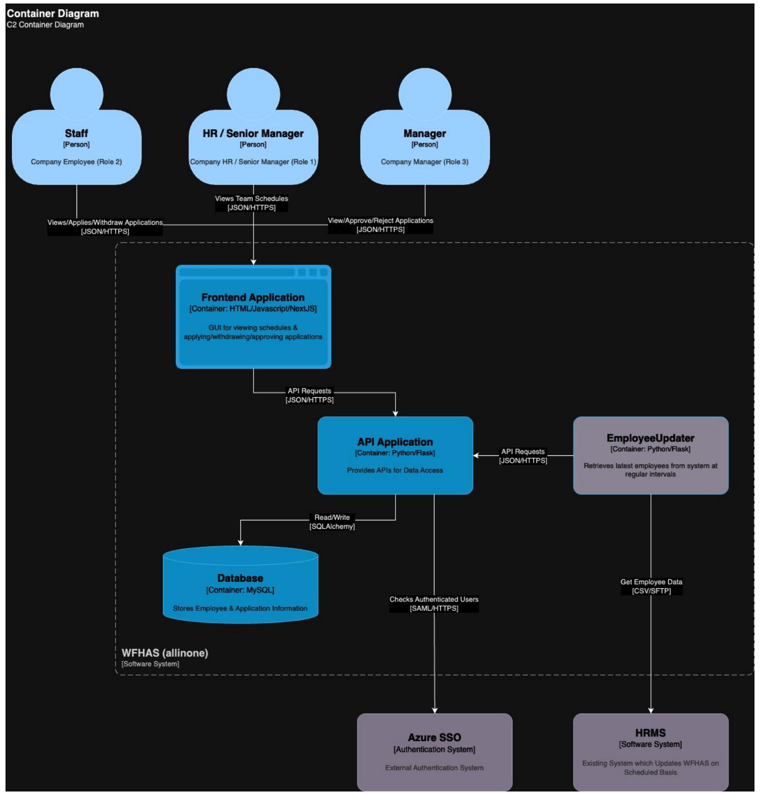
✓ Going through DoD Check

	Joel	Yew Kit	Benedict	Yan Jie	Kasen
Unit Tests Passed?	-	-	-	-	✓
Test Cases Passed?	✓	✓	✓	✓	✓
Code Reviewed?	✓ (pair 1)	✓ (pair 2)	✓ (pair 2)	✓ (pair 1)	-
Demonstration with Feedback	• user login could use some logo, looks very plain	• Currently default view of view own schedule is set to week, maybe month will look nicer	-	-	-
All in Agreement?	✓	✓	✓	✓	✓

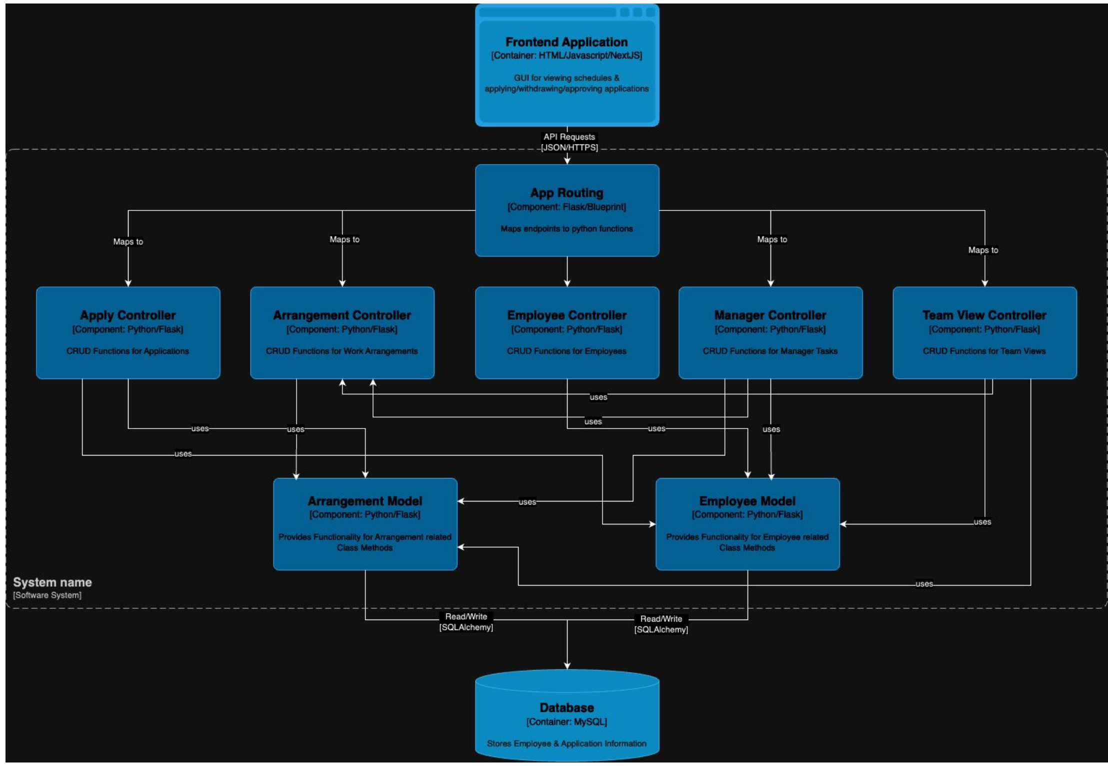
How we modelled / architected our system



C1


Container Diagram
C2 Container Diagram
Staff
[Person]
Company Employee (Role 2)
HR / Senior Manager
[Person]
Company HR / Senior Manager (Role 1)
Manager
[Person]
Company Manager (Role 3)
Views Team Schedules
[JSON/HTTPS]
Views/Applies/Withdraw Applications
[JSON/HTTPS]
View/Approve/Reject Applications
[JSON/HTTPS]
Frontend Application
[Container: HTML/Javascript/NextJS]
GUI for viewing schedules & applying/withdrawing/approving applications
API Requests
[JSON/HTTPS]
API Application
[Container: Python/Flask]
Provides APIs for Data Access
API Requests
[JSON/HTTPS]
EmployeeUpdater
[Container: Python/Flask]
Retrieves latest employees from system at regular intervals
Read/Write
[SQLAlchemy]
Database
[Container: MySQL]
Stores Employee & Application Information
Checks Authenticated Users
[SAML/HTTPS]
Get Employee Data
[CSV/SFTP]
WFHAS (allinone)
[Software System]
Azure SSO
[Authentication System]
External Authentication System
HRMS
[Software System]
Existing System which Updates WFHAS on Scheduled Basis

C2


Frontend Application
[Container: HTML/Javascript/NextJS]
GUI for viewing schedules & applying/withdrawing/approving applications
API Requests
[JSON/HTTPS]
App Routing
[Component: Flask/Blueprint]
Maps endpoints to python functions
Maps to
Maps to
Maps to
Maps to
Apply Controller
[Component: Python/Flask]
CRUD Functions for Applications
Arrangement Controller
[Component: Python/Flask]
CRUD Functions for Work Arrangements
Employee Controller
[Component: Python/Flask]
CRUD Functions for Employees
Manager Controller
[Component: Python/Flask]
CRUD Functions for Manager Tasks
Team View Controller
[Component: Python/Flask]
CRUD Functions for Team Views
uses
Arrangement Model
[Component: Python/Flask]
Provides Functionality for Arrangement related Class Methods
Employee Model
[Component: Python/Flask]
Provides Functionality for Employee related Class Methods
System name
[Software System]
Read/Write
[SQLAlchemy]
Read/Write
[SQLAlchemy]
Database
[Container: MySQL]
Stores Employee & Application Information

C3

How we ensured a high-quality codebase

Organised & Modular Codebase

Model - View - Controller
- Backend developed in Flask
- Models manage data, controllers handle logic and API requests
- Streamlined backend

Reusable Components
- Frontend developed in Next.JS
- Leverage reusable components
- Reusable hooks, utility functions, use effects, for repeatable functions
- Lightweight and maintainable frontend

Cloud Deployment

Frontend
- Vercel
- Designed for Next.JS projects

Backend
- PythonAnywhere
- Extensive documentation and ease of use
- Flexible and configurable for many different functionalities

Database
- Amazon RDS on AWS
- Cost-effective, easy to configure

Environment Based Development

Environment Configuration Management
- Separate configurations for development and testing

Secure Secret Management
- AWS secrets manager
- GitHub secret HMAC encoding

Techniques we Employed

Trunk Based Development
- Small, frequent commits to your branch, integration only after testing

Pair Programming
- Each dev has a partner to work with
- Share knowledge, best practices

Regular Refactoring
- Reduce code bloat and other smells

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Computing and Information Systems

How we tested our system

Automated

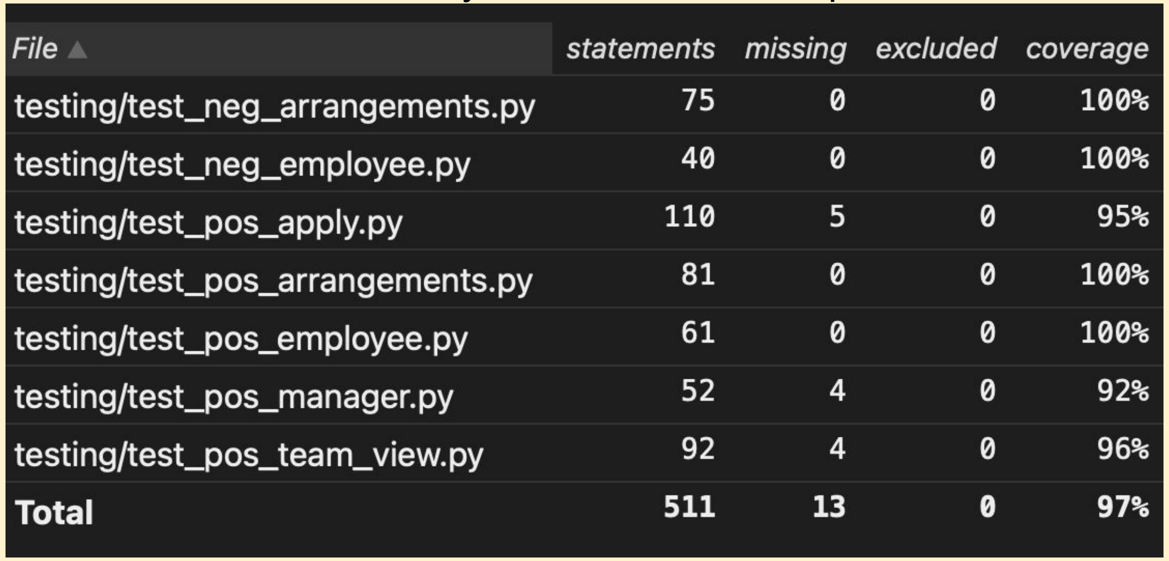
Pytest (Black Box Testing)

- Employed for unit testing and integration testing
- 37 test functions for 20 different API calls
- Test Coverage of **97%**

pytest

Optimisations for FIRST Testing Principles

- Fast testing using in memory database for testing configs
- Enabled timely test case writing using pytest-mock, testing APIs even when they have unfinished dependencies

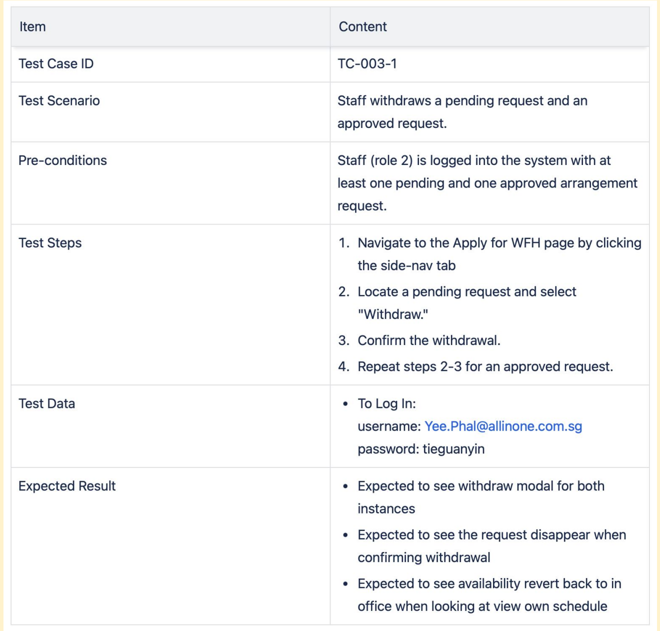


File ▲	statements	missing	excluded	coverage
testing/test_neg_arrangements.py	75	0	0	100%
testing/test_neg_employee.py	40	0	0	100%
testing/test_pos_apply.py	110	5	0	95%
testing/test_pos_arrangements.py	81	0	0	100%
testing/test_pos_employee.py	61	0	0	100%
testing/test_pos_manager.py	52	4	0	92%
testing/test_pos_team_view.py	92	4	0	96%
Total	**511**	**13**	**0**	**97%**

Manual

Detailed Test Cases (White Box Testing)

- Each user story, each acceptance criteria, has a documented test case in our confluence
- Followed test case inputs and expected outputs during manual testing
- Reviewed and confirmed all test cases as part of DOD



Item	Content
Test Case ID	TC-003-1
Test Scenario	Staff withdraws a pending request and an approved request.
Pre-conditions	Staff (role 2) is logged into the system with at least one pending and one approved arrangement request.
Test Steps	1. Navigate to the Apply for WFH page by clicking the side-nav tab 2. Locate a pending request and select "Withdraw." 3. Confirm the withdrawal. 4. Repeat steps 2-3 for an approved request.
Test Data	• To Log In: username: Yee.Phal@allinone.com.sg password: tieguanyin
Expected Result	• Expected to see withdraw modal for both instances • Expected to see the request disappear when confirming withdrawal • Expected to see availability revert back to in office when looking at view own schedule

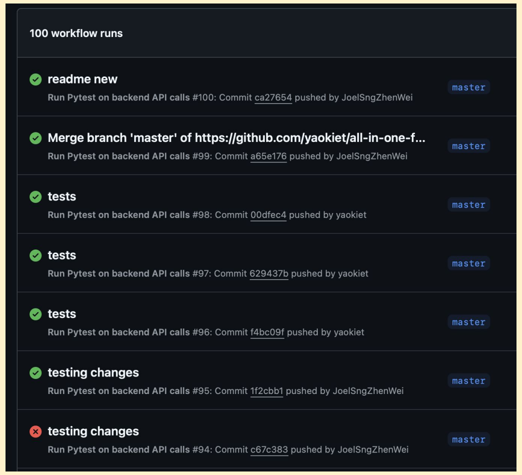
SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Computing and Information Systems

How our process was supported by CI/CD

Continuous Integration

Automated pytest using GitHub Actions

- Runs on every push for `backend` and `master` branches
- Covers unit testing on 'simple' controllers
- Covers integration testing on 'complex' controllers that make several internal API calls to other controllers
- Advantages
  - Easier to implement with better plugin support than unittest
  - Plugins availability: pytest-flask, pytest-mock, pytest-faker
- Challenges
  - Test coverage was hard to maintain as code base grew (Current is at 97%)
  - Complexity of testing and mocking for complex APIs that made several internal API calls



100 workflow runs

- ✓ **readme new** — Run Pytest on backend API calls #100: Commit ca27654 pushed by JoelSngZhenWei — `master`
- ✓ **Merge branch 'master' of https://github.com/yaokiet/all-in-one-f...** — Run Pytest on backend API calls #99: Commit a65e176 pushed by JoelSngZhenWei — `master`
- ✓ **tests** — Run Pytest on backend API calls #98: Commit 00dfec4 pushed by yaokiet — `master`
- ✓ **tests** — Run Pytest on backend API calls #97: Commit 629437b pushed by yaokiet — `master`
- ✓ **tests** — Run Pytest on backend API calls #96: Commit f4bc09f pushed by yaokiet — `master`
- ✓ **testing changes** — Run Pytest on backend API calls #95: Commit 1f2cbb1 pushed by JoelSngZhenWei — `master`
- ✗ **testing changes** — Run Pytest on backend API calls #94: Commit c67c383 pushed by JoelSngZhenWei — `master`

Continuous Deployment

Frontend via Vercel

- On push, Vercel automatically redeploys
- Advantages
  - Alerts if deployment unsuccessful
  - Detailed build logs, strict linting requirements, helps promote clean code development
- Challenges
  - Severless deployment, poses challenges with Flask session, secrets management

Backend via PythonAnywhere

- On push, GitHub issues webhook to a special api
- On receiving, PA runs a bash console command to git pull and git merge
- Advantages
  - Designed for seamless deployment of Flask backends
  - Detailed error, access, and server logs
  - Scheduling tasks functionality
- Challenges
  - Signature validating and security via secure webhook tokens, encoding with HMAC, validating signatures.