

Trackr - Iteration 3

Presentation / Demo

BU MET CS 673 (Summer 1 2022)

Group 1

- Jean Dorancy
- Timothy Flucker
- Xiaobing Hou
- Weijie Liang

Outline

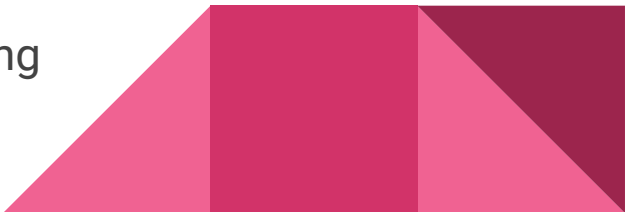
- Role Assignments
- Project Overview
- Requirement Analysis
- Project Architecture
- Implementation
- Testing
- Security
- Deployment
- Project Management
- Demo



Role Assignments

Team Member Name	Role(s)
Jean Dorancy	Team Leader
Timothy Flucker	Design / Implementation Leader + Requirements Leader
Xiaobing Hou	Security Leader + Configuration Leader
Weijie Liang	QA Leader

High Level Project Overview

- **Description:** Trackr is a an application that is meant to be an alternative to financial tracking applications such as Mint and Truebill.
 - **Goal:** A way for users to view transactions against their bank accounts so that they can understand their overall financial well-being and spending behavior.
 - **Implementation:**
 - Spring Boot Application for REST APIs
 - React and React-bootstrap for the frontend
 - Deployed on Heroku as Docker container
 - Postman for manual and JUnit for automated testing
- 

Requirements Analysis

During Iteration 0, an initial set of requirements was determined and recorded in the SPPP. These requirements were broken down into two major categories with the following sub-categories:

- Functional Requirements
 - Essential - User Management APIs, Bank Account Management APIs, Transaction Management APIs, Web GUI
 - Desirable - Web GUI Password Change
 - Optional - N/A
- Non-Functional Requirements - Swagger Document, JWT Token + Filter, CSRF protection

The stories were then organized into the following epics based on their related resource:

- User Management
- Bank Account Management
- Transaction Management

At the beginning of Iteration 1, stories were scoped for each iteration and assigned. At the beginning of each iteration, the stories were re-examined and re-assigned based on team needs/limitations.

- As project evolved, stories were added to Pivotal Tracker to track new development / configuration.

Project Functionality

- User Management

- Register
- Login and Logout
- View User Profile
- Update User Profile
- Change Password

- Bank Account

- Create New Bank Account
- Update Bank Account
- Invalidate Bank Account
- Find All User's Bank Accounts
- Find User Bank Account By ID

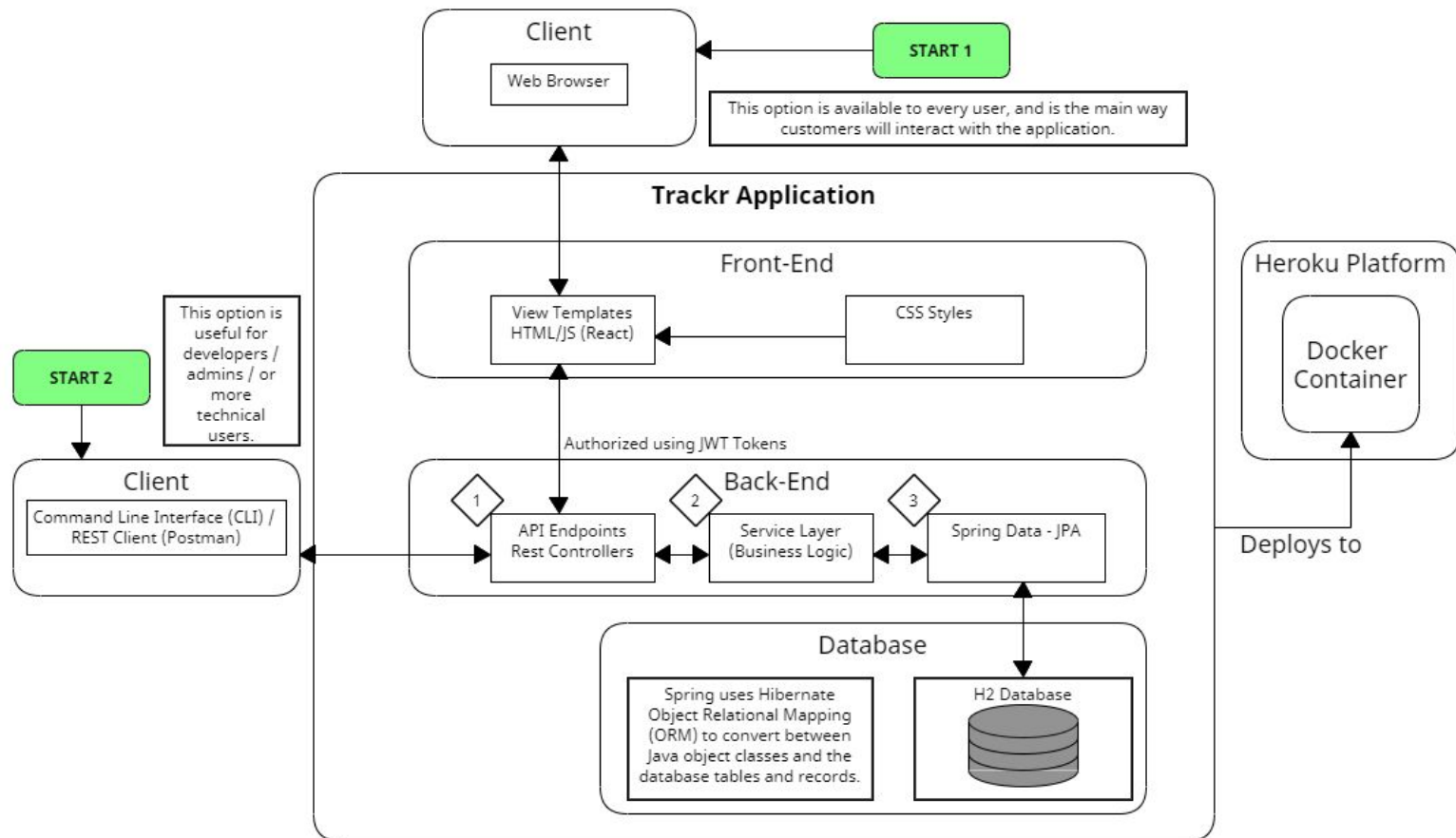
- Transaction Management

- Create New Transaction
- Modify Transaction
- Invalidate Transaction
- Find All User's Transactions
- Find Transaction By ID

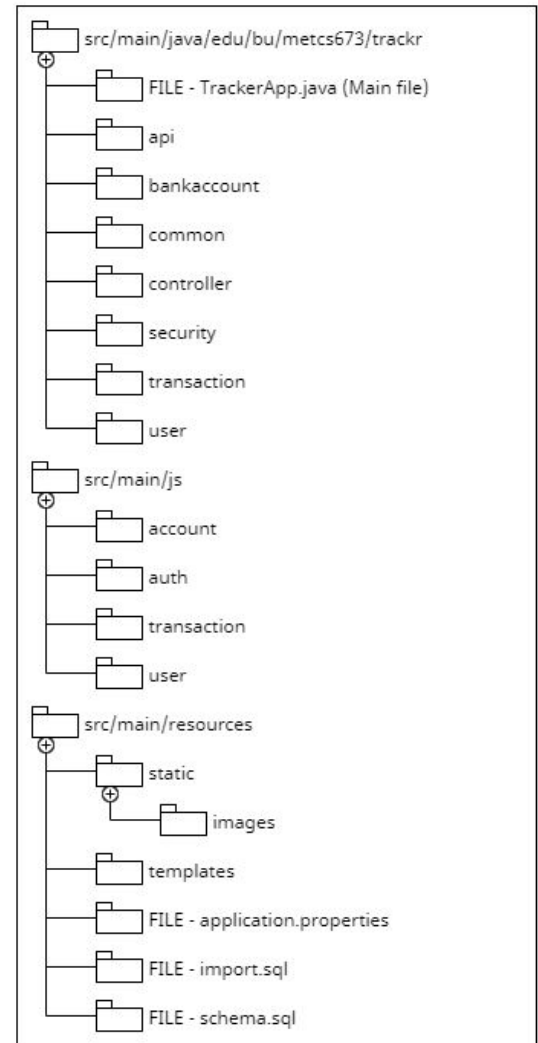
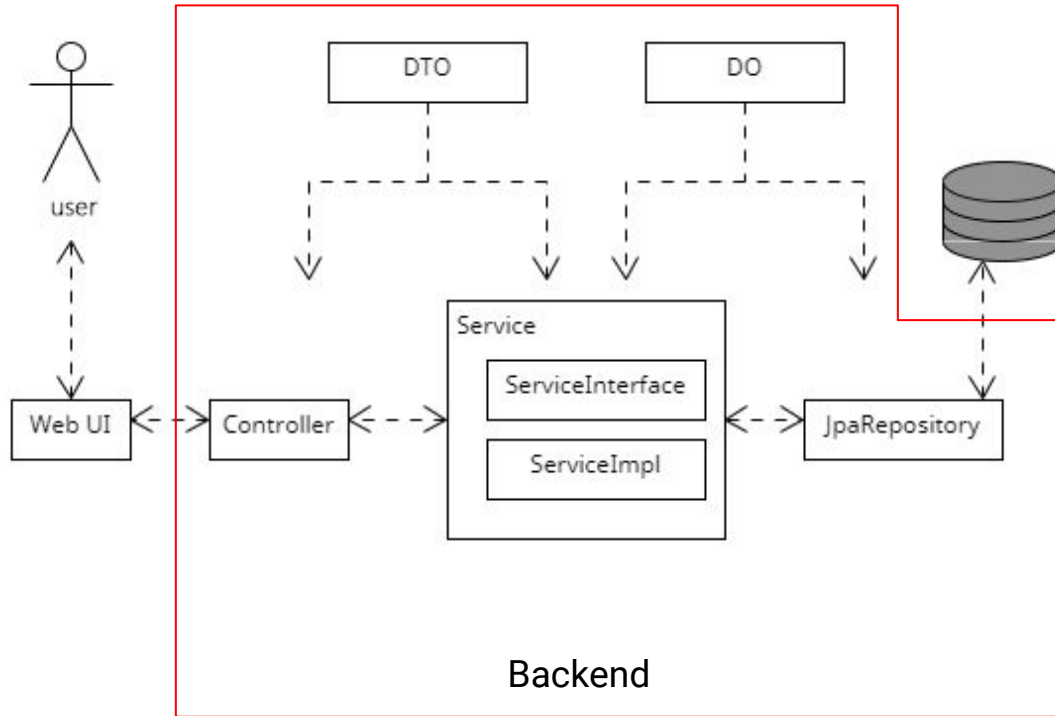
- User Dashboard

- Bank Accounts Preview
- List of all Transactions
 - Across all accounts
 - Ordered by Date DESC

Project Architecture



Implementation: Project Structure



Implementation: Frontend


React Container Pattern.

- **Single Responsibility Principle:** Components that are focused on one thing.
- **Service Class for API:** Abstract away the backend API.
- **Container Component:** Data fetching and renders presentation components.
- **Presentation:** Simply showing the data and inputs to the user as desired.



Implementation: Backend

APIs with layered architecture with distinct classes for each resource:


- **Controller:** Exposes REST API endpoint.
 - **Service:** Interface which defines methods for a service.
 - **ServiceImpl:** Implementation of a service interface.
 - **Repository:** JPA repository which interacts with the Database.
 - **Entity Class:** Java representation of database table with validation.
 - **Data Transfer Object (DTO):** Presentation object for API request/response.
- 

Testing Overview

Automated Unit Testing

- Backend
- Frontend

Manual Integration Testing

- Create User Account
 - Login and Logout
 - Create, Edit, and Delete Bank Account
 - Create, Edit, and Delete Transaction
 - Edit User Profile, Change Password
- 

Automated Unit Testing

Backend

JUnit and Mockito

- 163 Test Cases
- 12 Test Classes

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 163, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.327 s
[INFO] Finished at: 2022-06-19T13:08:54-04:00
[INFO] -----
```

Frontend

Jest and React Testing Library

- 52 Test Cases
- 10 Test Suites

```
PASS src/main/js/account/__tests__/account.test.js
PASS src/main/js/auth/__tests__/login.test.js
PASS src/main/js/transaction/__tests__/transaction-form.test.js
PASS src/main/js/auth/__tests__/login-form.test.js
PASS src/main/js/account/__tests__/bank-account-form.test.js
PASS src/main/js/user/__tests__/profile-form.test.js
PASS src/main/js/user/__tests__/sign-up-form.test.js
```

```
Test Suites: 10 passed, 10 total
Tests:      52 passed, 52 total
Snapshots:  0 total
Time:       3.365 s
Ran all test suites matching /test/i.
```

Backend Unit Testing Coverage Report

Current scope: all classes

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	92.3% (24/26)	76.1% (118/155)	65.7% (255/388)

Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
edu.bu.metcs673.trackr	100% (1/1)	66.7% (2/3)	66.7% (2/3)
edu.bu.metcs673.trackr.api	100% (1/1)	63.6% (7/11)	50% (7/14)
edu.bu.metcs673.trackr.bankaccount	100% (6/6)	77.8% (28/36)	58.9% (43/73)
edu.bu.metcs673.trackr.common	66.7% (2/3)	62.5% (5/8)	63.6% (7/11)
edu.bu.metcs673.trackr.controller	100% (2/2)	42.9% (3/7)	23.1% (3/13)
edu.bu.metcs673.trackr.security	100% (3/3)	100% (13/13)	100% (76/76)
edu.bu.metcs673.trackr.transaction	100% (5/5)	83.8% (31/37)	70.1% (61/87)
edu.bu.metcs673.trackr.user	80% (4/5)	72.5% (29/40)	50.5% (56/111)

Frontend Unit Testing Coverage Report

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	76.34	58.69	56.09	76.08	
account	71.42	75	43.75	70.58	
account-preview.js	81.81	50	66.66	80	15,33
account.js	33.33	100	20	33.33	22-48
bank-account-form.js	100	100	100	100	
bank-account-modal.js	100	50	100	100	25
bank-account.js	25	100	0	25	11-40
confirmation-modal.js	66.66	50	50	66.66	35
auth	100	100	100	100	
login-form.js	100	100	100	100	
login.js	100	100	100	100	
logout.js	100	100	100	100	
transaction	66.66	50	46.15	66.66	
transaction-conf-modal.js	66.66	50	50	66.66	35
transaction-form.js	75	50	66.66	75	17-29,71
transaction-modal.js	100	50	100	100	19-27
transaction-preview.js	37.5	100	28.57	37.5	30,62-91
user	86.36	57.14	75	86.36	
profile-form.js	80	57.14	60	80	20,120-157
profile.js	100	100	100	100	
sign-up-form.js	100	100	100	100	

Test Suites: 10 passed, 10 total

Tests: 52 passed, 52 total

Manual Integration Testing

In order to do the Manual Test on the Project and record the test results, we used a **spreadsheet** to record each test case. So far, all 5 high level defect has been solved.

CS673_STD_TestCase_team1

File Edit View Insert Format Data Tools Extensions Help

Last edit was 1 minute ago

100% \$ % .0_ .00 123 Arial 10 B I S A

A1

Test case name:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Test case name:	New or Old:	Test item:	Test priority:	Dependencies:	Preconditions:	Input data:	Test steps:	Postconditions:	Expected output:	Actual output:	Pass or Fail:	Bug id/link:	Additional notes:
2	Create user account	New	Add a new user account	High	None	A user account information that has never been used before is used to successfully create a user	First name(WEIJIE); Last name(LIANG); Username(WEL157); Email address(weijiel@bu.edu); Password(123456)	1. Start the page by entering: https://trackr-dev.herokuapp.com/ 2. Fill in the input box with the appropriate information 3. Click "Sign Up!" Button 4. See if a new account is successfully created	The user is able to log in with the created account	Account successfully created!	Account successfully created!	Pass	None	None
3	Login	New	Use existing user information to login	High	None	user account needs to exist to test its successful login	Username(WEL157); Password(123456)	1. Start the page by entering: https://trackr-dev.herokuapp.com/ 2. Click the "Login" Button at the top of the page 3. Fill in the input box with the appropriate information 4. Click the "Login" Button 5. See if the account login successfully	The user login to their own account	Successfully authenticated!	Successfully authenticated!	Pass	None	None
4	Test invalid scenarios 1(Create Duplicate user names)	New	Test the corresponding function with a value that does not match the input conditions	High	None	A used user account information is used in order to successfully report an error	First name(WEIJIE); Last name(LIANG); Username(WEL157); Email address(weijiel@bu.edu); Password(123456)	1. After the former registration, click the "Home" button. 2. Fill in the input box with the appropriate information 3. Click "Sign Up!" Button 4. See if a new account is successfully created	The account should not be created	Invalid USERNAME value. Please use another value.	Invalid USERNAME value. Please use another value.	Pass	None	Also tested the case of different user names but the same mailbox, account added successfully, need to follow up to discuss whether to reject duplicate mailboxes
5	Test invalid scenarios 2(wrong login info)	New	Test the corresponding function with a value that does not match the input conditions	High	None	An incorrect user account information is used in order to successfully report an error	Username(WEL157); Password(987654)	1. Start the page by entering: https://trackr-dev.herokuapp.com/ 2. Click the "Login" Button at the top of the page 3. Fill in the input box with the appropriate information 4. Click the "Login" Button 5. See if the account login successfully	The authentication should not be passed	Invalid Login Credentials	Invalid Login Credentials	Pass	None	None

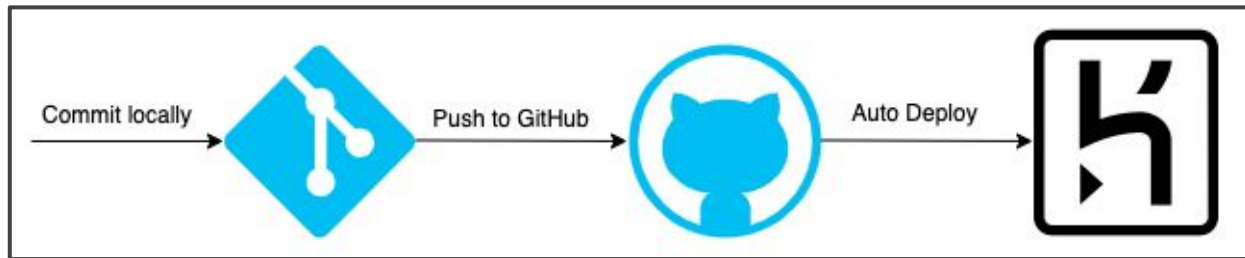
Security

- CIA (Confidentiality Integrity Availability) security model
- DAC (Discretionary Access Control) access-control method
- JWT (Json Web Token)
 - JWT token is validated first
 - TTL (time-to-live) of 15 minutes
- BCrypt Library (encode password)
- Integrated GitHub Workflows
 - Java Tests
 - JavaScript Tests
 - CodeQL Scans (security vulnerability)



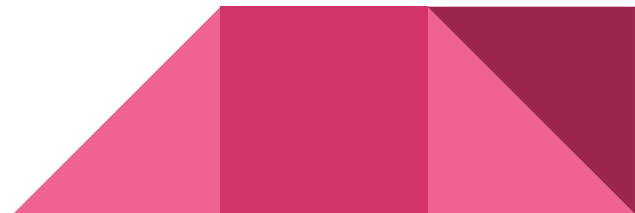
Deployment

- Commit Locally
- Push to GitHub
- Open PR
 - Checks
 - Java Tests
 - JS Tests
 - JS Code Sniff
 - Code Analysis with CodeQL
 - Java
 - JS
 - Feedback
 - Approval
- Deploy branch test with Heroku (manual)
- Merge (to **main** and **development** branches trigger auto deploy)
- App is deployed as a Docker container to Heroku



Project Management

- Risk Management
- Quality Management
- Last Iteration Development Completed
- Iteration Evolution
- Challenges and Lessons Learned



Risk Management

We identified a total 15 risks to the project below the ones with the highest priority.

Risk Title	Priority	Plan
Scope creep	16	Explore alternative solutions.
Lack of motivation or responsibility	15	Everyone to work on things they are interested in and practice Scrum rituals.
Unclear requirements	12	Constant communication with stakeholders.
Constant requirements change	8	Favor generic solutions that are adaptable and embrace change

Quality Management

- Code Reviews and Approval for PRs
- Deployment to Development
 - Before every merge
 - Automatically enforced by repository configuration
- GitHub Actions Checks
 - Java Tests
 - JavaScript Tests
 - JavaScript Code Style
 - Static Code Analysis with GitHub CodeQL (Java and JavaScript)
- Unit Testing
- Manual Integration Testing
 - Every developer as part of development workflow
 - QA Leader test at end of the iteration and generates test report



Last Iteration Development Completed

Frontend

- Change Password
- List of all Transactions DESC
- Unit Tests for UI Forms
 - Sign Up
 - Login
 - Profile
 - Bank Account
 - Transaction

Backend

- All Transactions For User API
- Unique Validation for Email
- Alphanumeric Validation
 - First Name
 - Last Name
- Refactor of Bank Account Entity to include list of Transactions.

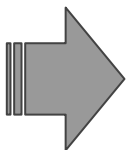
Configuration and Security

- CSRF Protection
- Code Analysis with CodeQL
 - Vulnerability scanning
 - Both Java and JavaScript
 - Run as pipelines on PRs
- JavaScript Tests Pipeline
 - Run all JavaScript tests
 - Only triggers on JavaScript change

Iteration Evolution

** Story Points are based on a Fibonacci sequence (Ex. 1,2,3,5,8) **

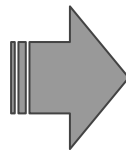
Iteration 1



22 Story Points

- Learning
- User API
- Bank Account API
- Home and registration Page
- Login Page
- Transaction API
- Deployment to Heroku with Docker

Iteration 2



44 Story Points

- Backend Refactoring
- Dashboard Page
- All Accounts Page
- Profile Page
- Logout
- Completed Bank Account API
- Java tests pipeline

Iteration 3

29 Story Points


- APIs Refactor to be RESTful
- Change Password
- List of all Transactions DESC
- UI Forms Unit tests
- JS Unit Tests Pipeline
- Code Analysis Pipeline

Challenges and Lessons Learned

Challenges

- Everyone on the team is more backend focus
- Learning enough React quickly to be productive
- Learning about frontend testing
- Student work unusable
- Task reassignments during Iteration 2
- A big time commitment

Lessons Learned

- Always plan ahead and check progress
 - Avoid sharing a Git branch with other developers
 - Always makes sure everyone knows what they are working on
- 

Demo Content

- Home Page
 - Home
 - Register
- Login
- Dashboard Page
 - Create transaction
 - Edit transaction
 - Delete transaction
- Accounts Page
 - Create account
 - Edit account
 - Delete account
- Profile Page
 - Edit user information
 - Change password
- Logout



Let's figure out together where your money goes.



trackr

Thank You!

First name

Last name

Username

Email address

Password

Sign Up!