Trackr - Iteration 2
Presentation / Demo

BU MET CS 673 (Summer 1 2022)

Group 1
- Jean Dorancy
- Timothy Flucker
- Xiaobing Hou
- Weijie Liang

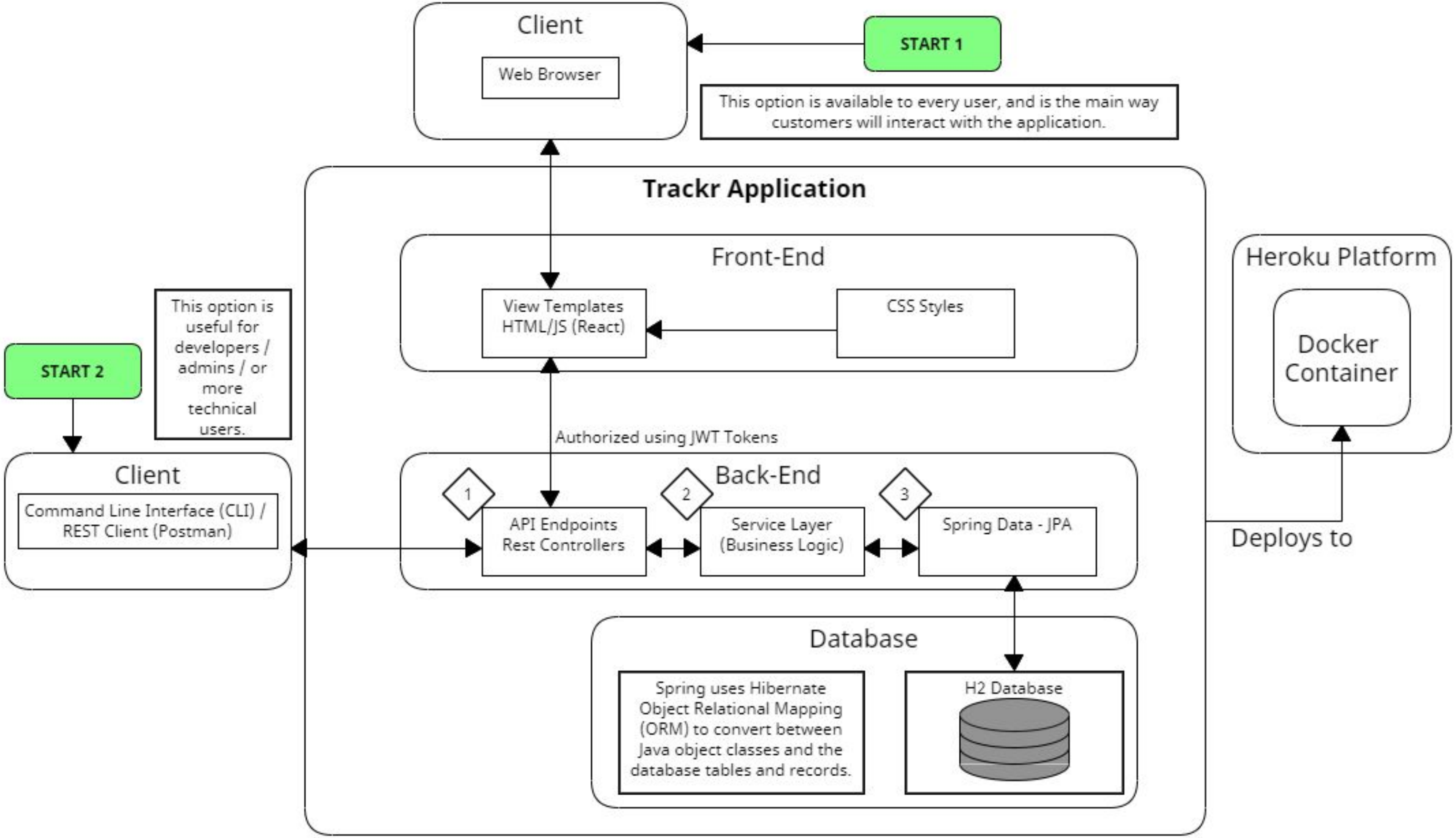
Outline

- Project Overview
- Project Architecture
- Development Completed for Iteration 2
- Testing Overview
- Unit Testing Coverage Report
- Deployment Update
- Next Steps

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

Project Architecture



Client
Web Browser
START 1
Trackr Application
Front-End
View Templates HTML/JS (React)
CSS Styles
Authorized using JWT Tokens
Back-End
1
2
3
API Endpoints Rest Controllers
Service Layer (Business Logic)
Spring Data - JPA
Database
H2 Database
Heroku Platform
Docker Container
Deploys to
START 2
Client
Command Line Interface (CLI) / REST Client (Postman)

This option is available to every user, and is the main way customers will interact with the application.

This option is useful for developers / admins / or more technical users.

Spring uses Hibernate Object Relational Mapping (ORM) to convert between Java object classes and the database tables and records.

Development Completed Iteration 2

Frontend

- Dashboard Page
  - Accounts
    - List of all accounts
  - Transactions
    - List of transactions
    - Create transaction
    - Edit transaction
    - Delete transaction
- All Accounts Page
  - List of accounts
  - Create account
  - Edit account
  - Deactivate account
- Profile Page
  - Edit user information
- Logout

Backend

- Refactor to Package by Feature
- Refactor API Paths to be RESTful
- Refactor API Response Object
- Refactor Duplicate code
- Updated JWT filter to skip unsecured paths
  - Registration
  - Login/Logout
  - Assets and Images

Configuration and Security

- Configuration
  - Pipeline to run Java tests on PR
  - Refactor to use Node JS Docker container in build
- Session Management
  - Spring Boot
    - **HTTP only** cookie for JWT
    - User logged in state cookie
  - React
    - Only checks user logged state cookie

Testing Overview

Automated

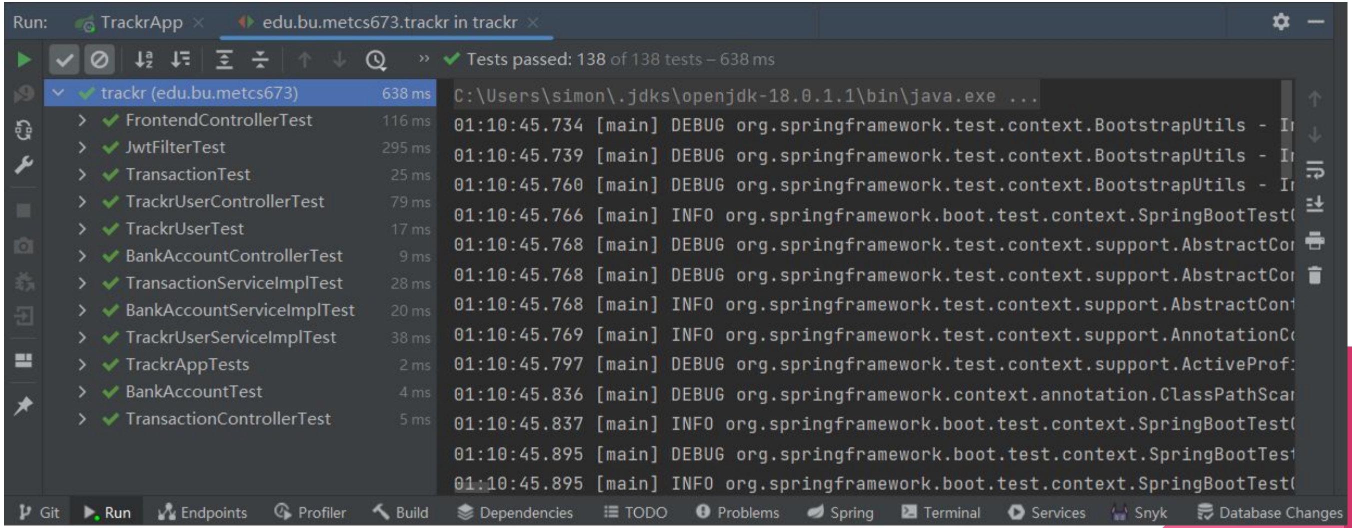
- JUnit5 and Mockito used for Unit tests
    - 138 tests
    - 12 test classes

Manual

- Postman
    - Shared Postman Collection (saved in the project repository)
- Web GUI
    - Create User Account
    - Login and Logout
    - Create, Edit, and Delete Bank Account
    - Create, Edit, and Delete Transaction
    - Edit Profile

Unit Testing

- To test our API, we use the JUnit and Mockito to write our automation tests. There are 138 automation tests in 12 test classes completed so far and they are geared toward the Bank Account, User, and Transaction aspects of the functionality.

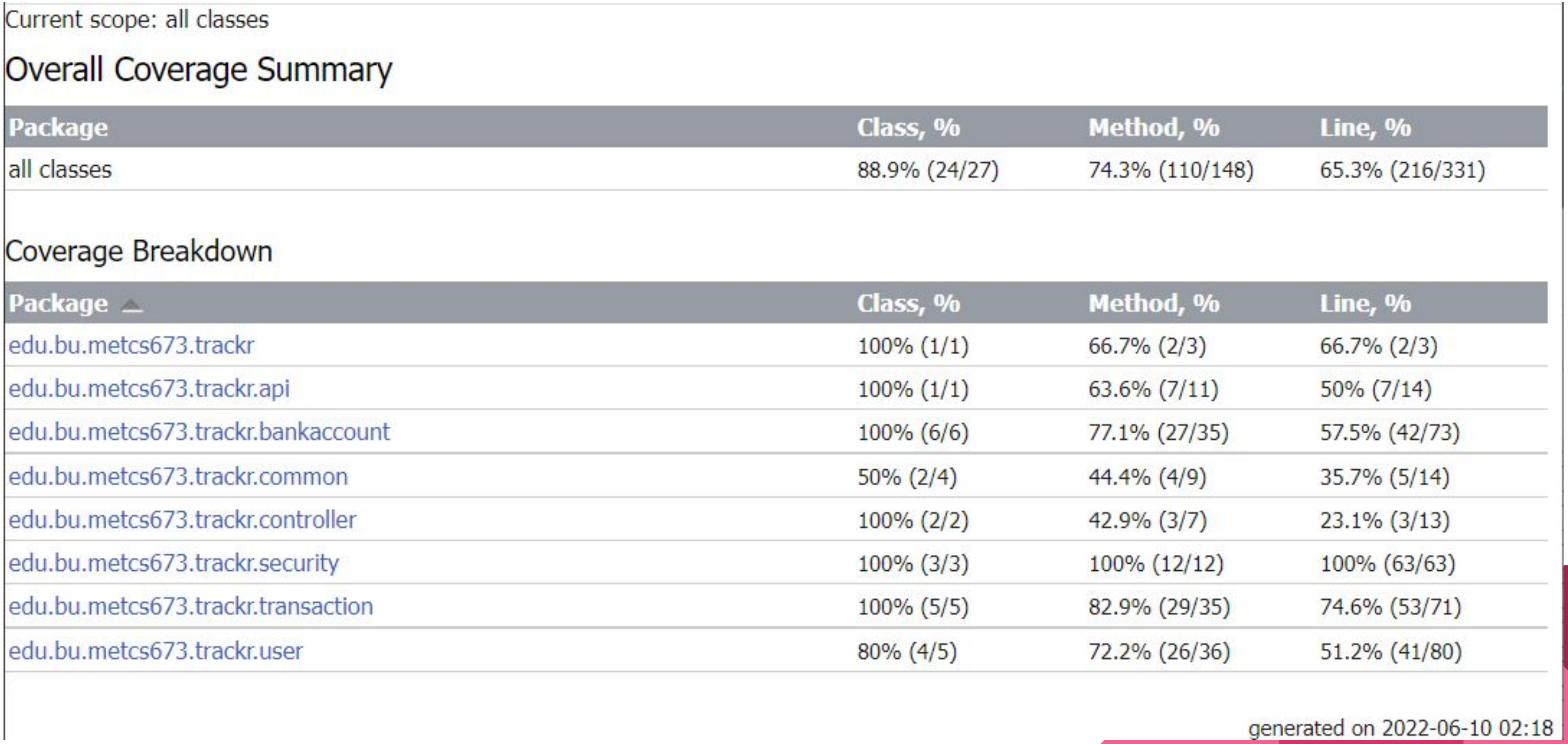


Tests passed: 138 of 138 tests – 638 ms

```
trackr (edu.bu.metcs673)        638 ms
  FrontendControllerTest        116 ms
  JwtFilterTest                 295 ms
  TransactionTest                25 ms
  TrackrUserControllerTest       79 ms
  TrackrUserTest                 17 ms
  BankAccountControllerTest       9 ms
  TransactionServiceImplTest     28 ms
  BankAccountServiceImplTest     20 ms
  TrackrUserServiceImplTest      38 ms
  TrackrAppTests                  2 ms
  BankAccountTest                 4 ms
  TransactionControllerTest       5 ms
```

```
C:\Users\simon\.jdks\openjdk-18.0.1.1\bin\java.exe ...
01:10:45.734 [main] DEBUG org.springframework.test.context.BootstrapUtils - Ir
01:10:45.739 [main] DEBUG org.springframework.test.context.BootstrapUtils - Ir
01:10:45.760 [main] DEBUG org.springframework.test.context.BootstrapUtils - Ir
01:10:45.766 [main] INFO org.springframework.boot.test.context.SpringBootTest(
01:10:45.768 [main] DEBUG org.springframework.test.context.support.AbstractCor
01:10:45.768 [main] DEBUG org.springframework.test.context.support.AbstractCor
01:10:45.768 [main] INFO org.springframework.test.context.support.AbstractCont
01:10:45.769 [main] INFO org.springframework.test.context.support.AnnotationCo
01:10:45.797 [main] DEBUG org.springframework.test.context.support.ActiveProf:
01:10:45.836 [main] DEBUG org.springframework.context.annotation.ClassPathScar
01:10:45.837 [main] INFO org.springframework.boot.test.context.SpringBootTest(
01:10:45.895 [main] DEBUG org.springframework.boot.test.context.SpringBootTest
01:10:45.895 [main] INFO org.springframework.boot.test.context.SpringBootTest(
```

Unit Testing Coverage Report



Current scope: all classes

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	88.9% (24/27)	74.3% (110/148)	65.3% (216/331)

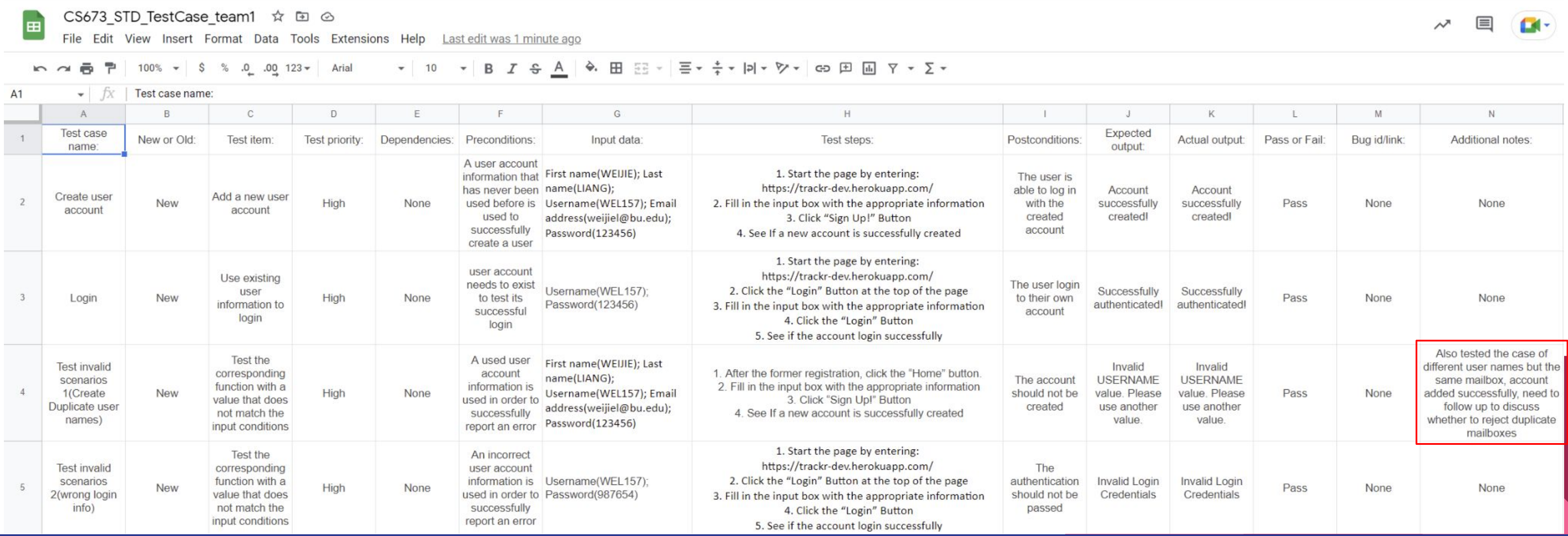
Coverage Breakdown

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)	77.1% (27/35)	57.5% (42/73)
edu.bu.metcs673.trackr.common	50% (2/4)	44.4% (4/9)	35.7% (5/14)
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% (12/12)	100% (63/63)
edu.bu.metcs673.trackr.transaction	100% (5/5)	82.9% (29/35)	74.6% (53/71)
edu.bu.metcs673.trackr.user	80% (4/5)	72.2% (26/36)	51.2% (41/80)

generated on 2022-06-10 02:18

Manual 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, until the last test, the developed features generally passed the Manual Test.

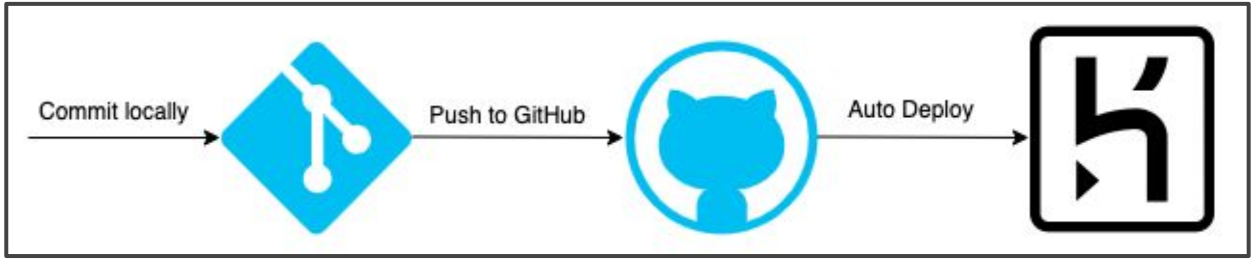


Test case name:

	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

Deployment Update

- Commit locally
- Push to GitHub
- Open PR
  - Checks
    - **Java Test**
    - JS Code Sniff
  - 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

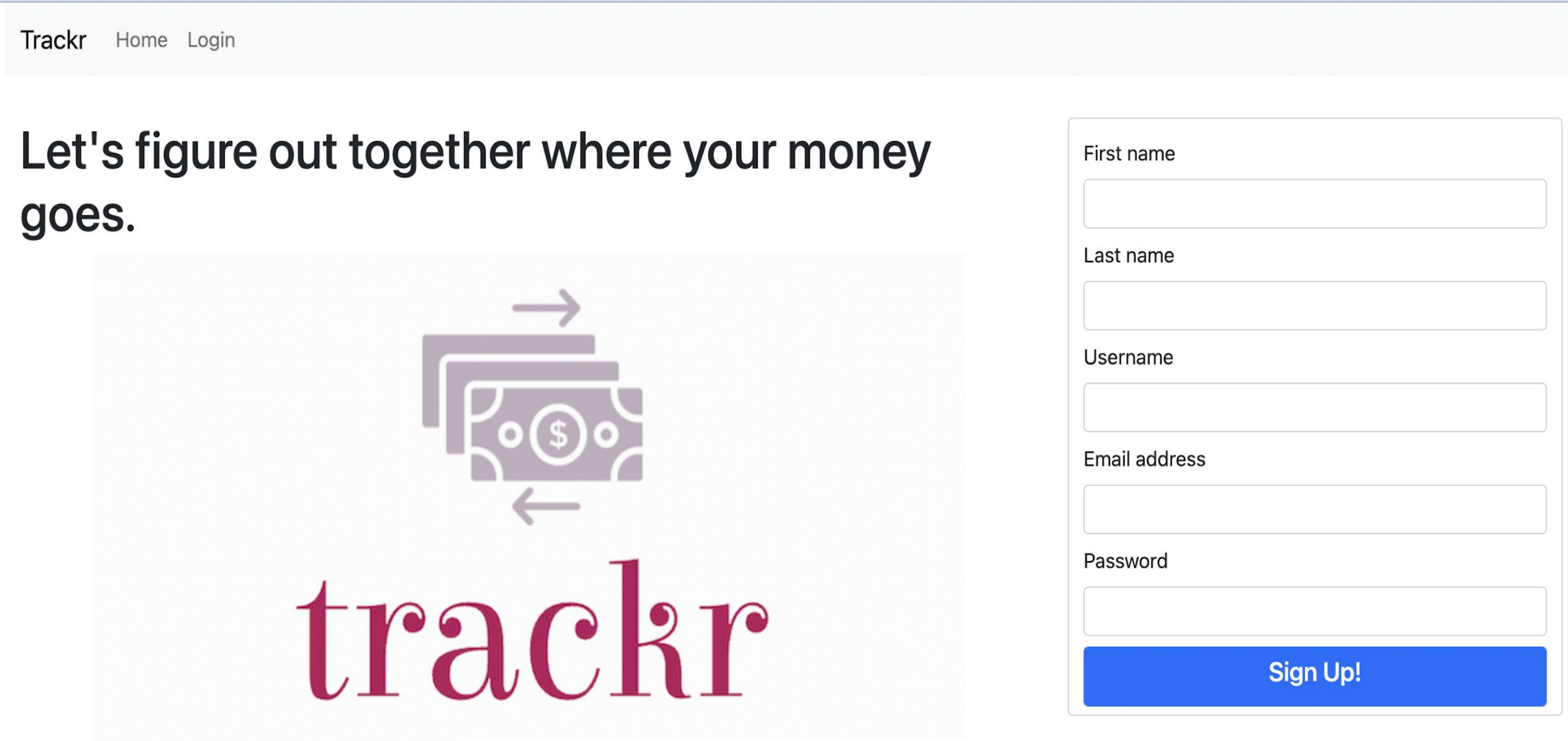


Next Steps

- Refactor the BankAccount class (`Add List<Transaction>attribute`)
- Display all the transactions across all accounts
- Change password
- Pipeline for running SonarQube checks for Java (optional)
- Pipeline for running JavaScript tests (optional)
- Implement Integration tests for UI (optional)

Demo Content

- Dashboard Page
  - Create transaction
  - Edit transaction
  - Delete transaction
  - Switch account
- Accounts Page
  - Create account
  - Edit account
  - Delete account
- Profile Page
  - Edit user information
- Logout



Let's figure out together where your money goes.

First name

Last name

Username

Email address

Password

Thank You!