

Project Waterfall

by *TEAM WoofWoof*

[Presentation Slides](#)

Joseph Hilsberg | z3470057

Quoc Anh Hoang | z5029533

Stephanie Anne Chua | z5076472

Lean Lynn Oh | z5144464

Panashe Mutema | z5075561

Project Waterfall

[Overview](#)

[Allocation of Roles](#)

[Meeting Schedule](#)

[Product Backlog Epics & User Stories](#)

[Final Release Features](#)

[Relevant Technologies](#)

[Programming Practices](#)

[System Setup](#)

[Technical Documentation](#)

[User Documentation](#)

[Initial Design Mocks](#)

[Backend Database Schema](#)

[Known Issues](#)

Overview

Project Waterfall is a web application that enables easy payments, bill splitting and monitoring of recurring expenses between individuals and groups of people. Implementation will only use in-app currency as **proof-of-concept**.

Users can pay, request and split money amounts with each other via in-app currency balance, as well as set up groups for common funds. Each user and group of users will have a special TrickleID to identify payments going to and from such parties. Recurring and late payments as well as low account credit will result in an email notification. The main user experience comes in the form of a dashboard with various views for users to manage future and current payments.

Allocation of Roles

The team has democratically decided to separate the main roles into frontend and backend developers who will work on different aspects of the project. The Scrum Master is involved in

ensuring both teams communicate and integrate their work together, and the Product Owner will see to it that the acceptance criteria of each user story is met.

<i>Team Member</i>	<i>Role</i>
Lean Lynn Oh	Frontend Developer
Panashe Mutema	Frontend Developer & Product Owner
Stephanie Chua	Developer & Scrum Master
Joseph Hilsberg	Backend Developer
Martin Hoang	Backend Developer

Meeting Schedule

Quick Stand-up : Every Tuesday, Thursday & Sunday

Official Meeting: Extra day on weekend (Fri/Sat/Sun)

- Sprint Planning Fortnightly
 - Features/bugs/chores to take upon
 - Delegation of tasks to different team members
- Sprint Review Fortnightly
 - What was achieved during the sprint & demos
 - What obstacles/bugs yet to be achieved
- Sprint Retrospective Fortnightly after Sprint Review
 - What went well
 - What went badly
 - Actions to be taken to improve

Product Backlog Epics & User Stories

- Each user story and epic has been inserted into Pivotal Tracker Management Tool
- The Product Backlog, the Icebox Product Backlog column has been prioritised in order; (first user story has the highest priority)

Final Release Features

- Homepage Website
- User Registration & Login
- Profile settings - change avatar, edit details
- Dashboard view for all transactions with search filters
- Account Balance & Account Management
- Single/Recurring Payments & Requests, and approvals and cancellations
- Pay & Request multiple users
- Group Creation & Management

- Group dashboard for group balance management & group transactions
- Email notifications on various user actions
- Interactive tutorials for those who have yet to use the system

Relevant Technologies

- Database Technologies
 - SQLite
- Cloud Servers & Backend
 - Heroku
 - Amazon Web Server
 - Python Django
 - SQL Alchemy
 - Jinja
- Frontend
 - JavaScript
 - SweetAlerts
 - Bootstrap Framework
 - Material UI Framework
- Testing Frameworks
 - Python unittest

Programming Practices

- Pair Programming

The team plans to use pair programming practices to collaborate different aspects of the codebase - eg between a frontend and backend developer. Furthermore as each team member has different backgrounds and experiences pair programming will provide a great way to share understanding of framework knowledge. Additionally, the team will use GitHub pull requests to provide code reviews before new features and code are merged to the master branch.
- Test-driven Development

The team has agreed to perform test-driven development involving some use of unit tests and user testing. At the end of each sprint review each feature and bug should be tested by different members of the team to ensure acceptance criteria was achieved.

 - Week 12 User Testing - gave the website to a few friends and family to test the user interface
 - Restructured sidebar
 - Decided to produce more page tutorials

System Setup

The following instructions are also available on the attached README.

Setup Development Environment

Update/install virtualenv using pip:

```
pip install virtualenv
virtualenv venv -p python3.6
```

Add this line to set email password for WaterfallPay, at the bottom of venv/bin/activate:

```
export WEBAPP_EMAIL_PASSWORD="email-password-here"
```

** Please request one of the team members for the email password*

Run the development environment:

```
source venv/bin/activate
```

```
pip install -r requirements.txt
```

Reload the Database with Dummy Data:

```
chmod +x reload_db.sh
```

```
./reload_db.sh
```

Run the website on localhost:

Run the server and open localhost on a web browser:

```
python manage.py runserver
```

Run custom unit tests:

```
python manage.py test webapp
```

- Note that this will sometimes have some failures because of an official Django bug; read [Known Issues](#)

Run scheduled script that updates transactions:

```
python manage.py update
```

Technical Documentation

User Accounts

- User object
- Profile object
 - Avatar links for images
- Account object
 - Balance is the sum of all transactions involving the account

Group Account

- Name - ID used to identify the group in transfers
- Account - storing the group's common fund
- Profiles of its member users
 - All users can add delete members from the group
 - All users can also access and use the group common fund

Deposits & Withdrawals Implementation

- Each Deposit/Withdrawal is a Transaction object
- Transaction objects contain the relevant Account, description, the type (withdrawal/deposit), the amount, date created and modified, as well as confirmation date
- Amounts are negative for withdrawals and positive for deposits
- Deleted transactions have `is_deleted` set true

Payments & Requests Implementation

- Each payment/request is a Transfer object consisting of two Transactions.

- Transfer has a Boolean is_request identifying Requests
- Requests' deadline refer to a date when it will be automatically rejected and deleted by the background process
- Payments' deadline refers to a future payment and when it will be executed. If there are insufficient funds, it will fail, be deleted
- Recurring transfer payments will create a copy of the recurring transfer set in the future each time it is confirmed
- Deleted transfers have is_deleted true

Background Process

A background process is meant to run sequentially on the server:

- Confirms recurring payments
- Deletes requests beyond the deadline and automatic payments with insufficient funds
- Sends relevant notifications for the above tasks

Email Notification Triggers

Notifications are sent to both parties, and notifications involving groups are not active.

All email notifications are sent on a separate thread to reduce any user-experienced lags.

- Approved Request
- Declined Request
- Cancelled Outgoing Payments/Requests
- Failed Recurring Payment/Request
- 3 Day Reminder for an Upcoming Payment/Request
- Low Balance - less than < \$10

User Documentation

The following describes the various features designed by the team and made available to users.

Landing Pages

- Home page allows user registration
- Teams and Product pages give information about the website

Registration

- New users can register from the homepage with a permanent username and an email

Main Dashboard

- Interactive tutorial is triggered if users have no transactions yet
- Contains a search filter
- Pending Transactions - which could be outgoing payments, outgoing requests, or received requests. Users can use the buttons here to approve or decline received requests, as well as cancel outgoing payments and requests.
- Past Transactions - shows a user's transaction history

Profile Settings

- Users can hover over their profile picture and change their display avatar
- Users can modify their first and last name, as well as change their password

Balance Management

- Users can view their balance, deposit and withdraw amounts (this is a proof of concept project, actual product will get this from a bank account)
- Users can also view their past deposits and withdrawals

TricklePay - Pay

- Tutorial triggered until user makes a payment
- Users can select whether to pay using their own balance or the common fund balance of the groups they belong to
- Users can select multiple users to pay
- The amount set for payment will be equal for all users selected as it does not make sense to 'split' a payment to users
- They can choose payment date in the future, recurring frequency, and give the payment a description. Payments set to today will be executed immediately. Future payments can be cancelled via the dashboard.

TricklePay - Request

- Tutorial triggered until user makes a request
- Users can select whether to make a request on behalf of themselves or the group they belong to
- Users can select multiple users to request money from
- The amount set for payment will be split amongst the selected users. Users can scroll down and decide the amounts each user will be requested to pay.
- They can choose a deadline date, recurring frequency, and give the request a description. Requests can be cancelled via the dashboard.

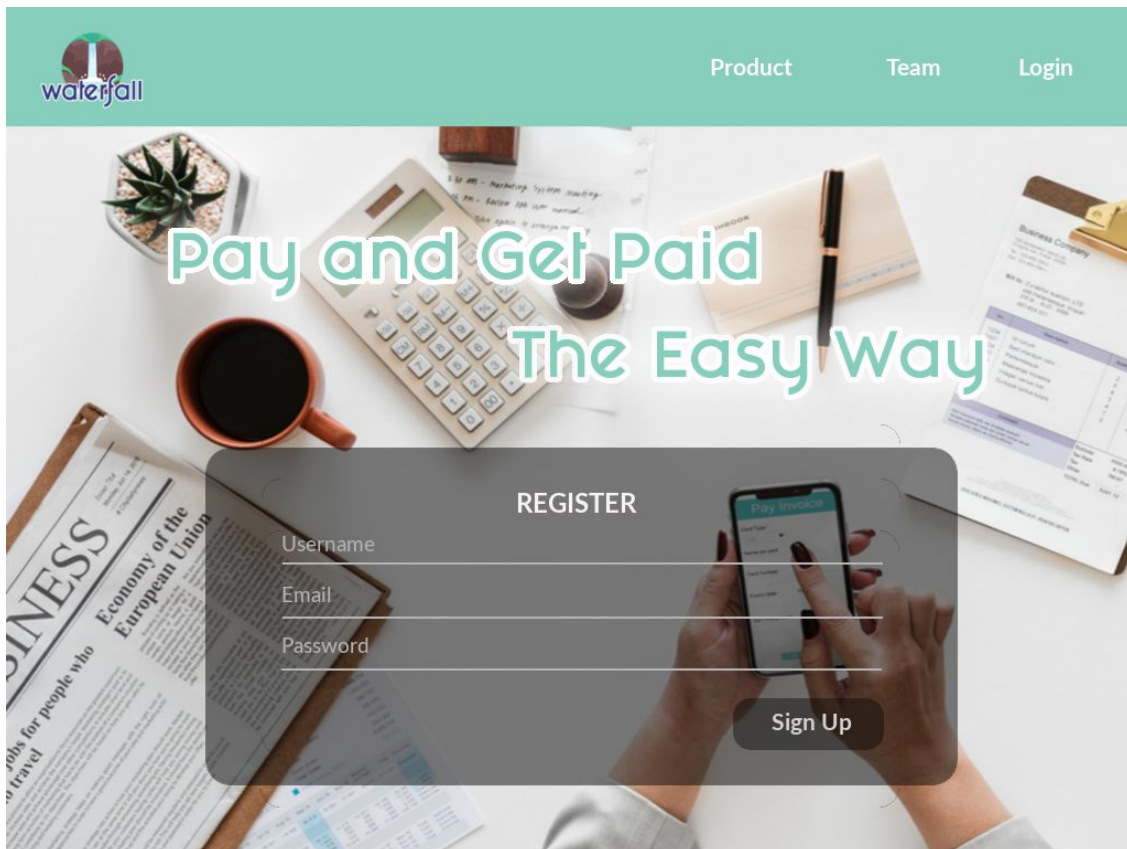
Group Management

- A list of groups the user belongs to is shown, alongside the group's balance, members and buttons to edit the group and access the group dashboard
 - **Group Dashboard**
 - User can view the group's balance
 - User can make deposits and withdrawals to the group from/to one's own balance
 - User can also view Pending Transactions and Past Transactions of the group, similar to that of the Main Dashboard. Similar rules apply; any user can approve or delete transactions
 - **Edit Group**
 - User can leave the group
 - User can add/remove new/current members of the group
- Users can also start the process of creating a new group
 - **Create New Group**


- User chooses a new group name which will also be used in all transfers involving the group
- User can add on different members via username
- Users can use the search filter to filter groups by name and member names


Initial Design Mocks

Homepage



Main Dashboard



BALANCE
\$52.35

Dashboard

Trickle Pay

Pay

Request

Split

Groups

Create Group

Search

Filter

Incoming Payments

View More

Name	Amount
@joseph	\$ 100.25
team.woof.woof	\$ 10.00

Outgoing Payments

View More


Name	Amount
@panashe	\$ 25.50
@martin_h	\$ 75.00


Past Payments

View More

Name	Amount
@lynn.oh	\$ 99.25

Payment Screen



BALANCE
\$52.35

Dashboard

Trickle Pay

Pay

Request

Split

Groups

Create Group

TricklePay

PayRequestSplit

TricklePay ID

Search

@stephanie X @alicia.k X

Amount

\$

.00

Date

Once

Weekly

Monthly

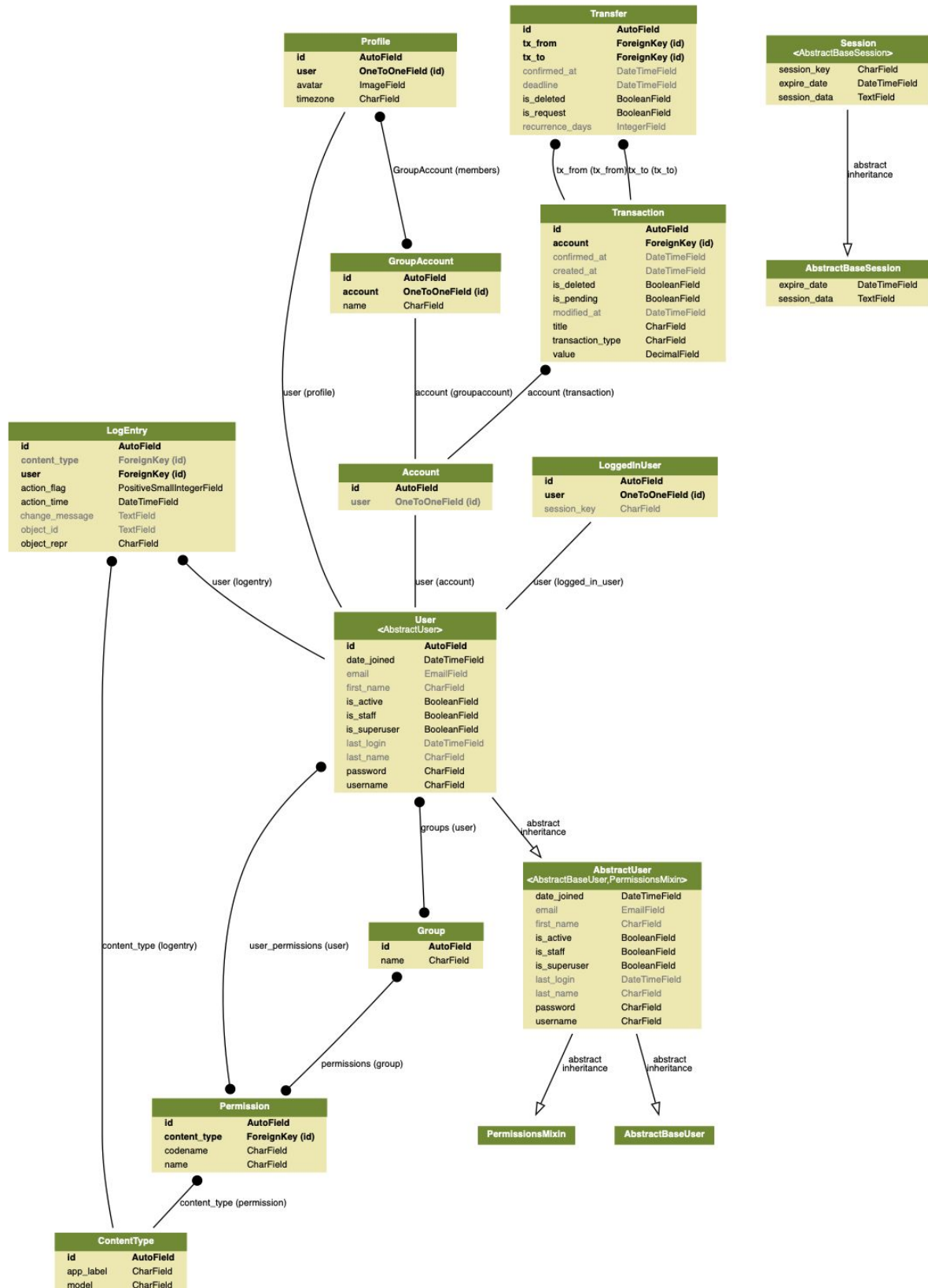
Yearly

Description

Type something...

TricklePay

Backend Database Schema



Known Issues

- [Django Unit Testing bug report](#)

Database lock errors might occur whilst running unit tests as a result of this bug report. Note that all tests do actually pass (0 errors) if the tests are repeated multiple times.