# Project Proposal – Financial Freedom

## Describe the proposal

‘Financial Freedom’ is a browser based web-application designed for individuals interested in a simple and easy way of managing their personal finances. It provides a variety of budgeting features, visual analytics, and different avenues of notifications that will assist and remind the user to keep track of their spending and savings.

The system allows users to create transactions which will be compared to the budgets and saving goals they have specified. It also gives users insight into how their own financial position compares to their past performance as well as others in the community.

The system provides visual analytics into users spending and saving habits, which can be organised and filtered based on a variety of criteria’s such as categories, time period, and amount.

Furthermore, ‘Financial Freedom’, will give users the opportunity to opt in to a notification system which will send emails, and SMS messages in response to unsustainable or irregular spending.

## Use cases

Create and edit budgets and saving goals

Create, edit, update, and delete ad-hoc income and expenses.

View spending analytics based on transaction type, demographic, time period, and amount

Compare personal finance to those within the community

General periodic reports of financial position of spending habits in a commerce style layout.

Opt in or out of public data pool.

Opt in or out of a notification system for irregular spending habits via SMS or email.

Update and follow other profiles

Theme app.

## Communication topology

The system will be implemented as a 3-tier architecture. This will include a web-client written on the React framework, using MobX for state management, and semantic-ui for styled components. The web server will be a python REST API on the falcon framework. Data will be stored in a PostgreSQL relational database. All interactions between the web server and the database will be done through the Peewee object relational mapper.

Example use case: Create budget

User fills in form, sends a put request to api\_url/budget, the web-server makes a transaction with the database, and on success, returns a HTTP\_OK (200) response, on failure throws an exception, and raises a HTTPError (500).

Example use case: Add transaction

Example use case: View analytics pages

## Design Mock-up