# CardSense

## Maximize Your Credit Card Rewards & Savings

CS180 Final Report

Xiyuan Wu, Andrew Do, Brandon Nguyen, Burhanuddin Taquee

# Table of Contents

# Overview

- Primary goal and functional objectives
- Development tools and technologies
- Libraries, frameworks, APIs

# Motivation

01

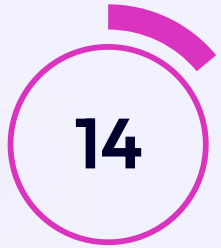
**Many Americans are paying high interest rates**

02

**Many Americans miss out on credit card rewards**


Can't afford their credit card bills


Can't pay their credit card bill in once




Credit card rewards unredeemed each year

# Functionalities

This project has 2 main functionalities:

1

### Track Spending

Track their spending of each transaction they made

2

### Maximize Rewards

Add their card to the app, and the app will recommend the best cards to use for different categories

# Tools & Technologies

**Frontend**

React, Tailwind CSS

**Backend**

Django

**Libraries & APIs**

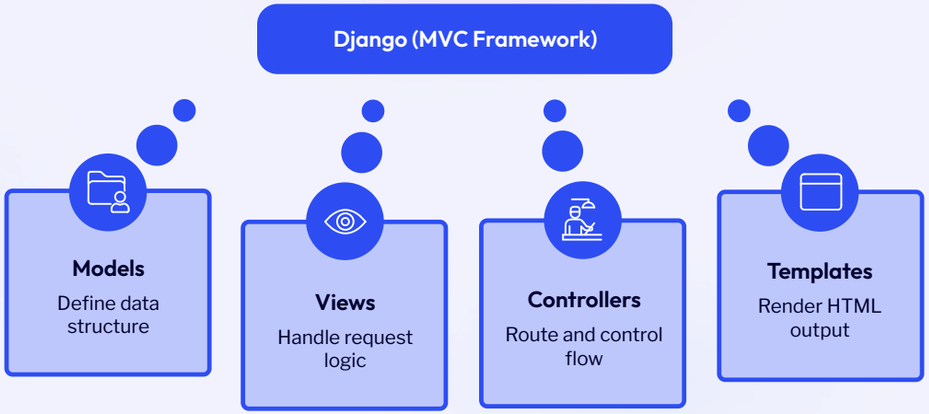
None. We build our own backend and write our APIs: REST API

# Design

- Overview of the architecture and diagram
- Design patterns
- Separation Design

# Architecture

## Design Pattern

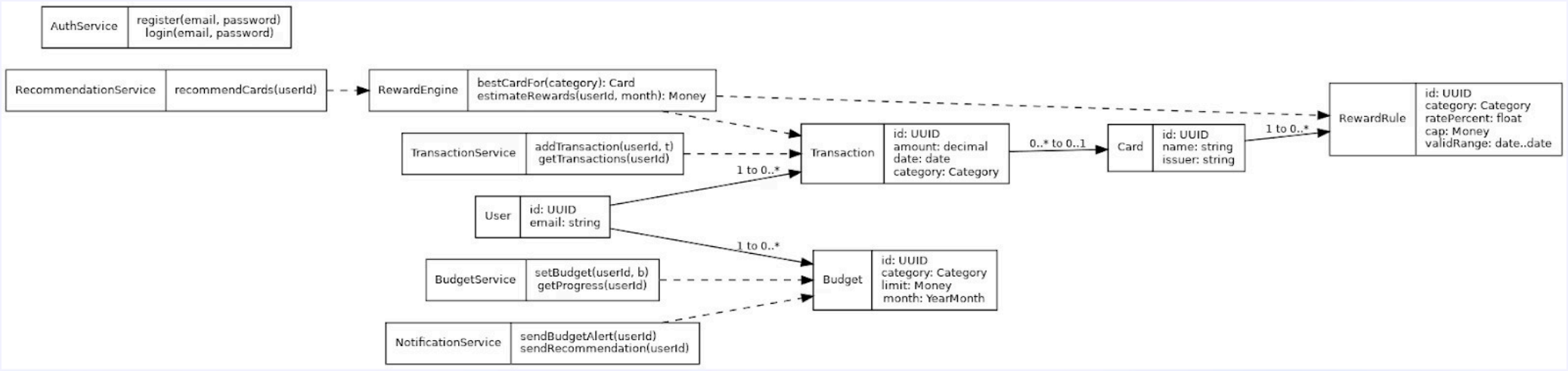



## Separation Design

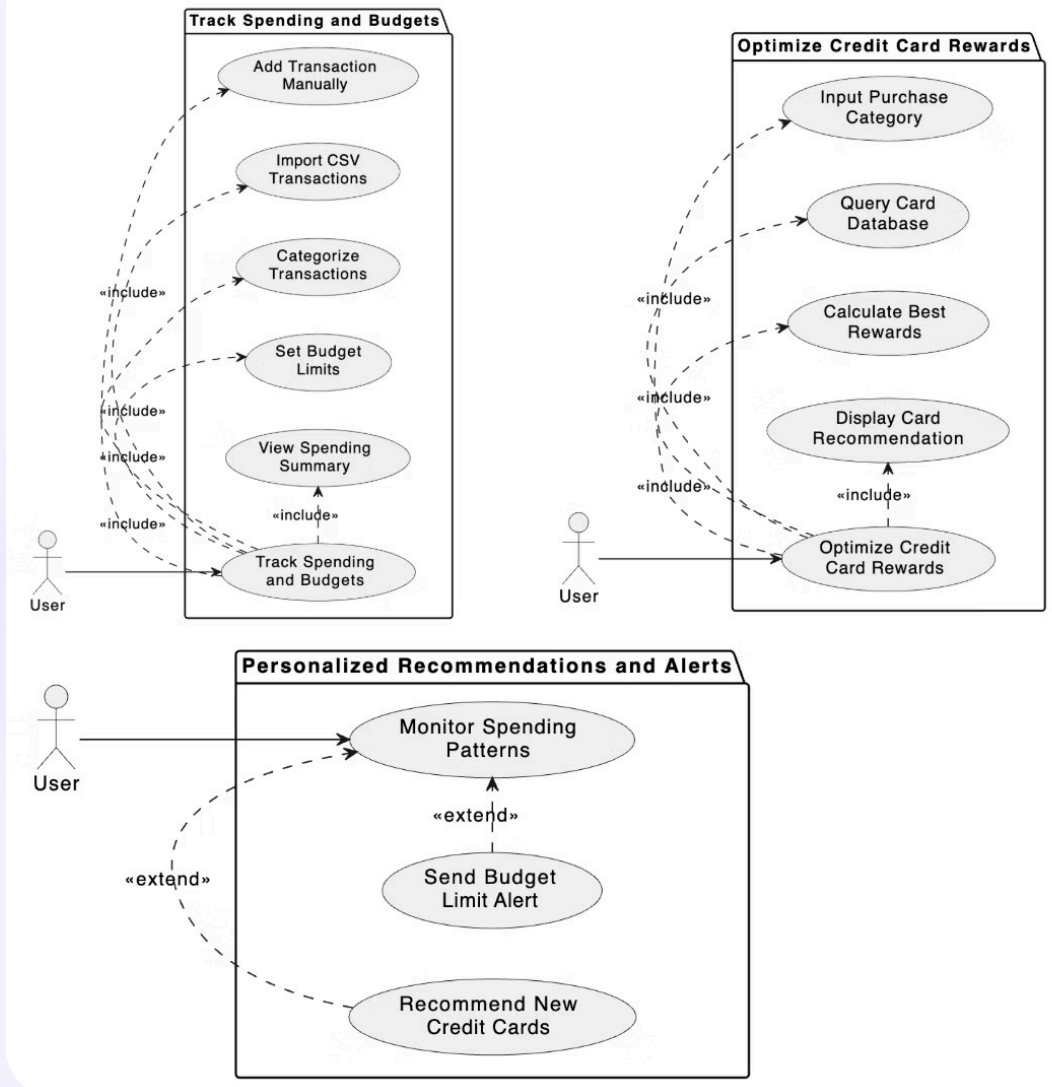
1. React + Django separation keeps UI fast and backend logic clean.
2. Modular apps (cards, budgets, transactions, optimizer) keep features organized and easy to expand.
3. REST API + signals ensure smooth data flow and real-time budget updates.
4. Architecture directly supports goals: track spending, manage budgets, and optimize card rewards.

# Diagram

## Class Diagram




## User Diagrams



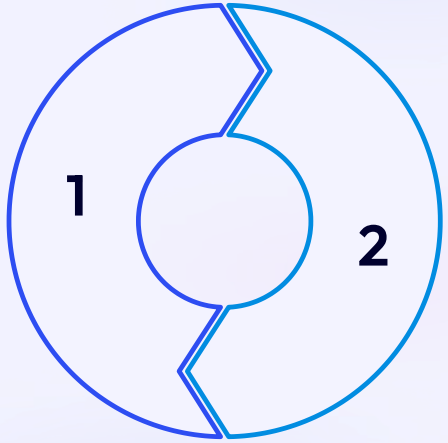
# Testing

- Testing frameworks
- Methodology: whitebox and blackbox

# Testing Frameworks

**Frontend**

Frontend use Jest and React Testing Library for testing




**Backend**

Backend use Python for testing, Django built-in test tool

# Whitebox

White-box testing verify whether the program exhibits a specific behavior.

1

### Frontend Whitebox

- Implemented test files using Jest and React Testing Library.
- Tested components, services, and utilities for correct behavior and user interactions.

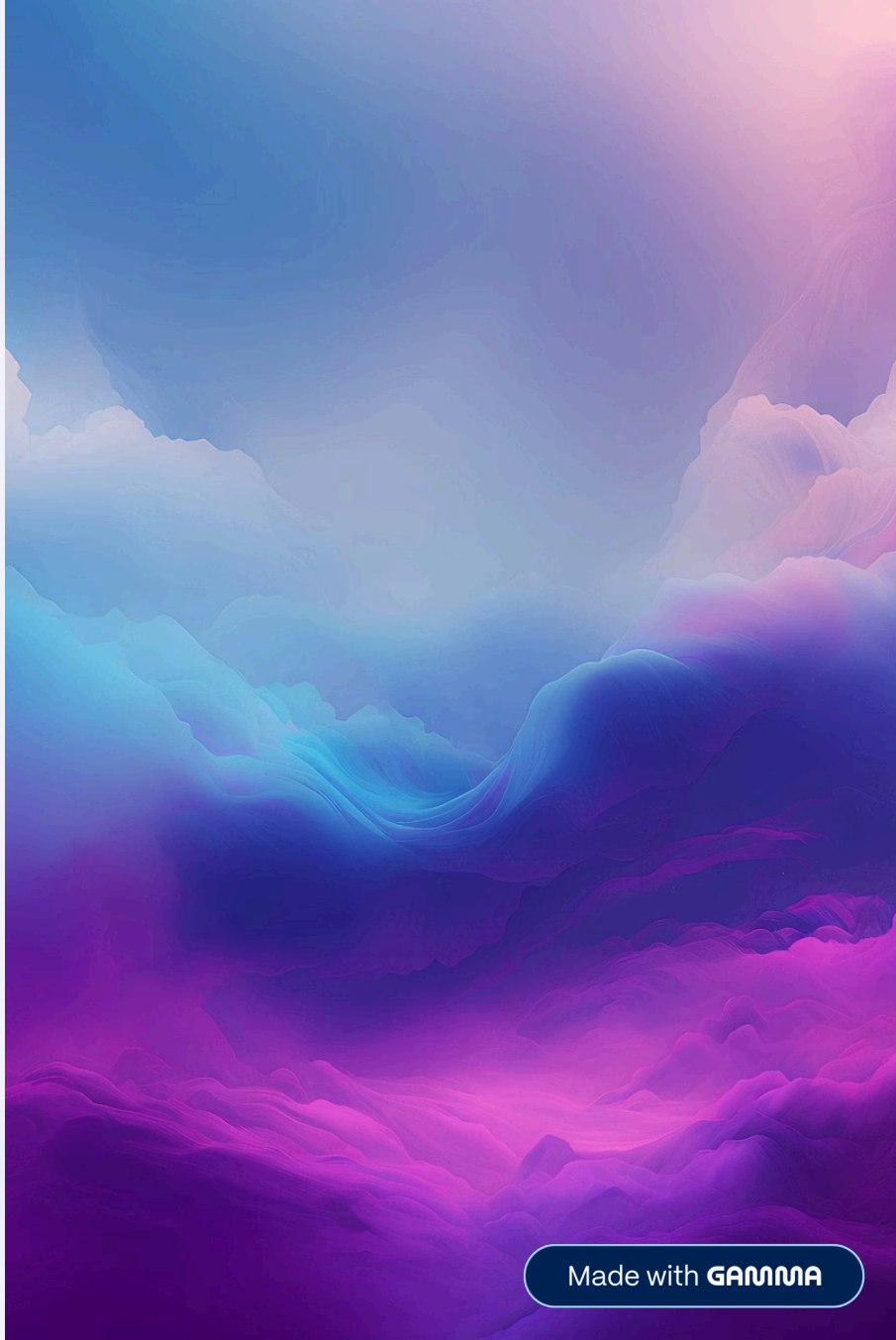
2

### Backend Whitebox

- Implemented Python test files under each Django app (accounts, transactions, budgets, cards, optimizer).
- Tested **models**, **serializers**, **permissions**, and **views** for correct behavior.
- All backend tests executed using Django's built-in test
  To run: python manage.py test "app name"

# Blackbox

- Simulated real user actions through the UI and API: sign-up, login, add cards, add transactions, import CSV, receive recommendations.
- Verified system behavior **without looking at internal code**: correct responses, error handling, permissions, and data validation.
- Ensured end-to-end flows worked as expected across frontend + backend.



# Testing Results

```
python manage.py test accounts
Found 31 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...............................
----------------------------------------------------------------------
Ran 31 tests in 2.076s

OK
Destroying test database for alias 'default'...
```

```
python manage.py test optimizer
Found 14 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..............
----------------------------------------------------------------------
Ran 14 tests in 1.361s

OK
Destroying test database for alias 'default'...
```

```
Found 28 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
........................
----------------------------------------------------------------------
Ran 28 tests in 2.481s

OK
Destroying test database for alias 'default'...
```

```
Found 14 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..............
----------------------------------------------------------------------
Ran 14 tests in 1.061s

OK
Destroying test database for alias 'default'...
```

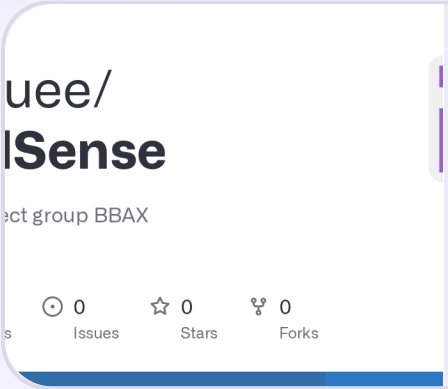
```
Found 29 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.............................
----------------------------------------------------------------------
Ran 29 tests in 10.328s

OK
Destroying test database for alias 'default'...
```

# Live Demo

Live Demo Time!

# Live Demo



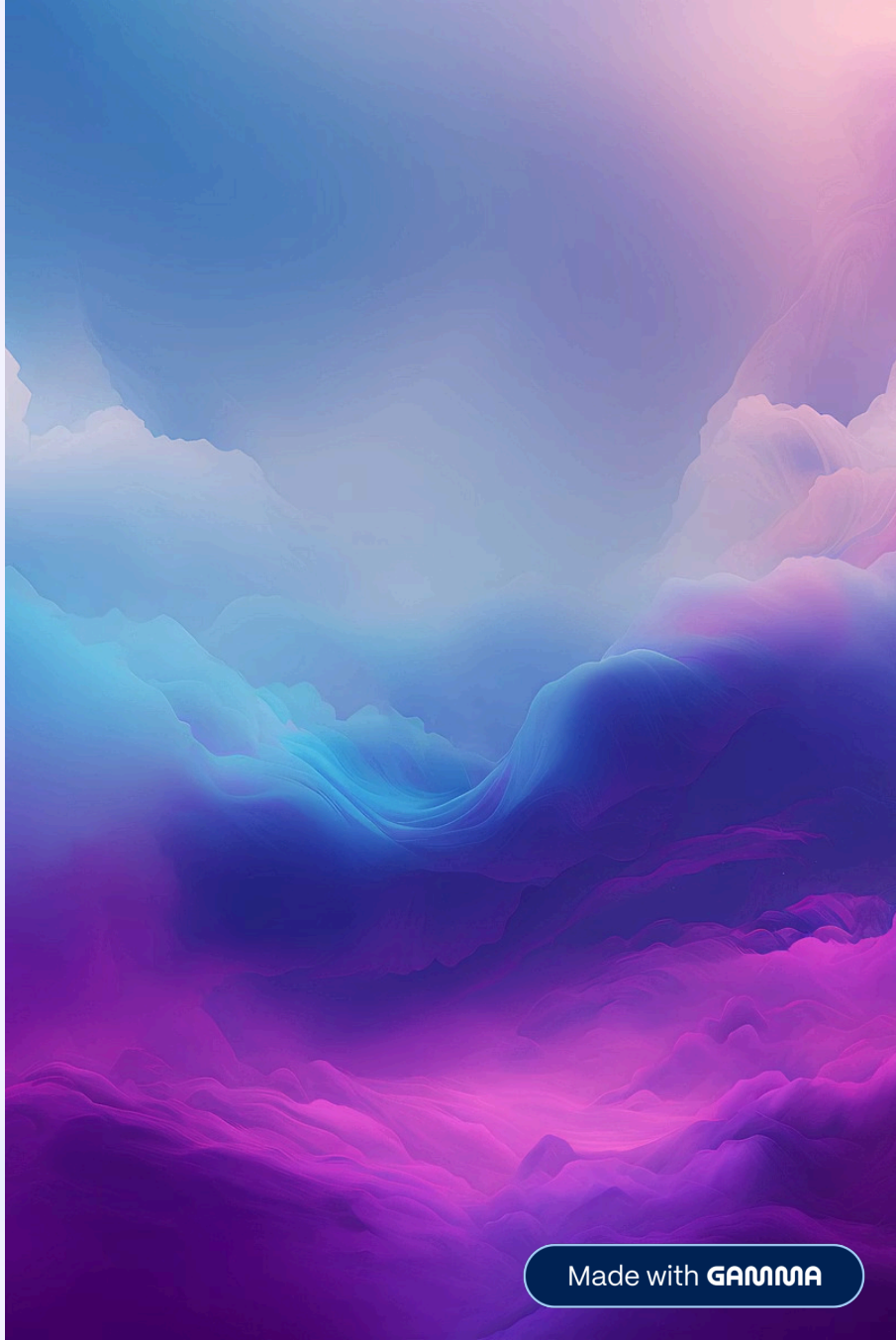

GitHub

**GitHub – btaquee/CardSense: CS180 ...**

CS180 Project group BBAX. Contribute to btaquee/CardSense development by...

# Challenges

1. **CSV import validation - handling bulk CSV imports**
    a. Did row by row processing and error-tracking
2. **Budget alert timing - needed to send budget alerts dynamically**
    a. Used Django signal to automatically recalculate
3. **Timezone handling for budgets - needed to account for user timezones**
    a. Implemented timezone conversion logic



# Any Questions?