Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design II
Spring 2020

logo.png

The Budgeteers
Budgeting App

Anil Karki
Robert Kemp
Emily Knowles
Michael Omordia
Roshan Shrestha

Revision History

Revision	Date	Author(s)	Description
0.1	11.11.2019	AK, BK, EK, HO, RS	document creation
0.2	12.11.2019	AK, BK, EK, HO, RS	document revision version 2.0

Contents

List of Figures

List of Tables

1 Introduction

The budgeting app will allow its users to monitor and analyze their spending habits in order to help them budget their money. The application itself will be a web app (possible a desktop application as well) that will allow a user to link a bank account. Once linked, the application will display transactions that the user makes using that bank account. This information will be used to help the user construct and maintain a budget. Multiple budgeting schemes will be available for the user to choose from, and they will be able to modify these schemes to their exact purposes.

Some key requirements of the application follow. The application must be secure. User accounts must be accessible only after user authentication. The user must be able to link a bank account via the Plaid banking API. The app should have multiple budgeting styles for the user to choose from. The application must be able to display a list of the user's recent transactions from Plaid. Using this data, the app should generate charts and visuals as well as make predictions and generate suggestions about the user's budgeting habits (whether or not they have met their goals), and other suggestions (such as perhaps where they could cut back to save money).

2 System Overview

Budgeting app consists of four layers: user layer, controller layer, api/server layer and database layer. The user layer interacts with user and collects information of user's preferences and send it to controller layer where it is processed and send to api/server layer. Api/server make requests to data base layer and obtained the information which is then sent back to controller layer.

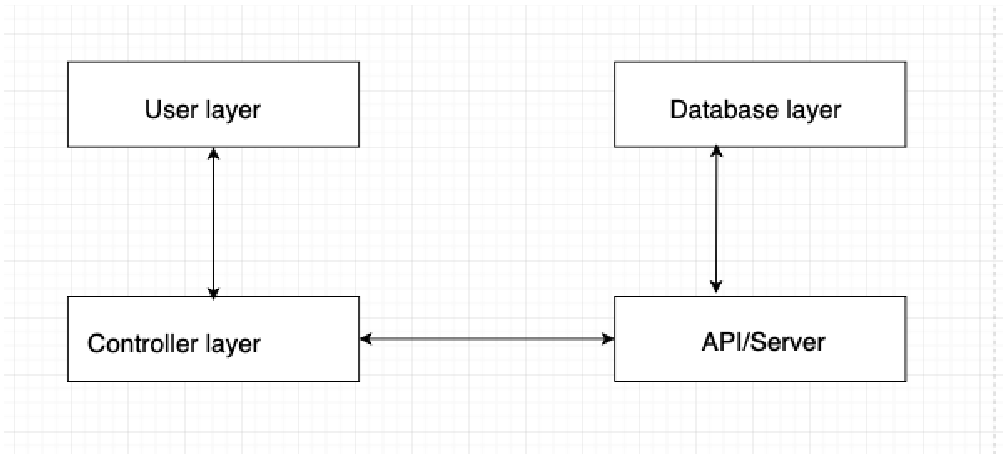

Figure 1: Budgeting app system overview

2.1 User Layer Description

The features of the client-side application include a login view, a home page view, and a settings view (encompassing various activities like mapping transaction names to buckets, determining budgeting style, etc.). This layer communicates with back end server and database via controller layer.

2.2 Controller Layer Description

This layer will communicate with the Application server, via HTTP requests (GET, POST, etc.). Basically, it acts like the middle man between server and client.

2.3 API Layer Description

The application server/API provides three main services to the client-side application: serving the actual web page that the client views, maintaining state of client applications through persistent database storage, and communicating directly with the Plaid API. All communications between the client application and the application server will take place in HTTP messages. JSON will be the form of encoding used to transmit data and state information, both between the client and app server, and the app server and the Plaid API.

2.4 Database Layer Description

The database layer will provide persistent and secure storage of all entities of the application. This includes user credentials, transactions, and user preferences.

3 Subsystem Definitions & Data Flow

Overall system is divided into four layers. User layer is further divided into three subsystem: registration, login and features. Controller layer has only one subsystem: app interface. Api layer has two subsystems: Django api and Plaid api and the database layer has only one subsystem: Django database. At first, user will use personal information to register, the user credentials will be passed django database subsystem with the help of app interface and django api subsystems. After registering, user can use user id and password to login the account. When user make request for login, controller layer's subsystem will made request to api/server layer which verify information with django database and return message of denial or acceptance to the controller layer.It will let user to add multiple bank account and get access to have benefits of multiple features. While adding bank account, django api subsystem sends and received data to and from plaid api, and stores in django database. Features subsystems will have various categories inside it. When user want to select the certain feature, controller layer made request to database layer and get back information with the help of api layer. This information will be passed to user layer which in turn shows to user as gui.

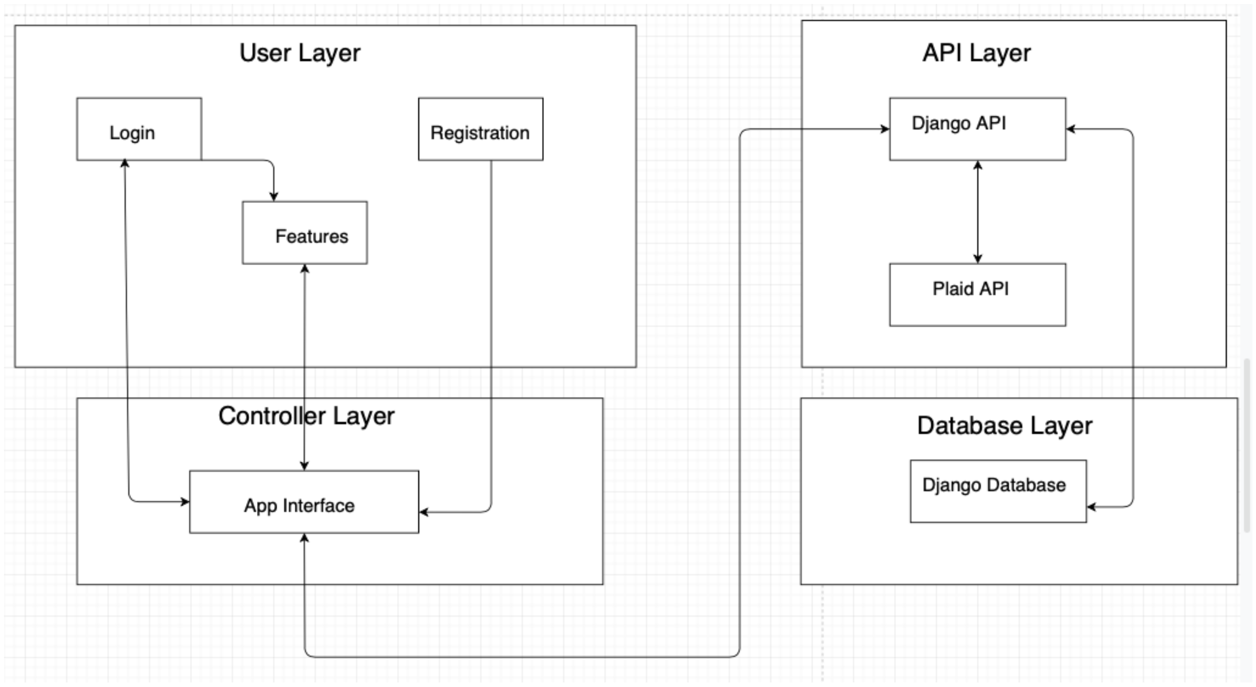

Figure 2: A simple data flow diagram

4 User Layer Subsystems

4.1 REGISTRATION

Registration being the First subsystem for User Layer; this is the primary step toward involving with the application. The user creates a new account by providing his/her credentials, which gets stored into system database. With the user being registered in the database, he/she can access the application features by logging in.

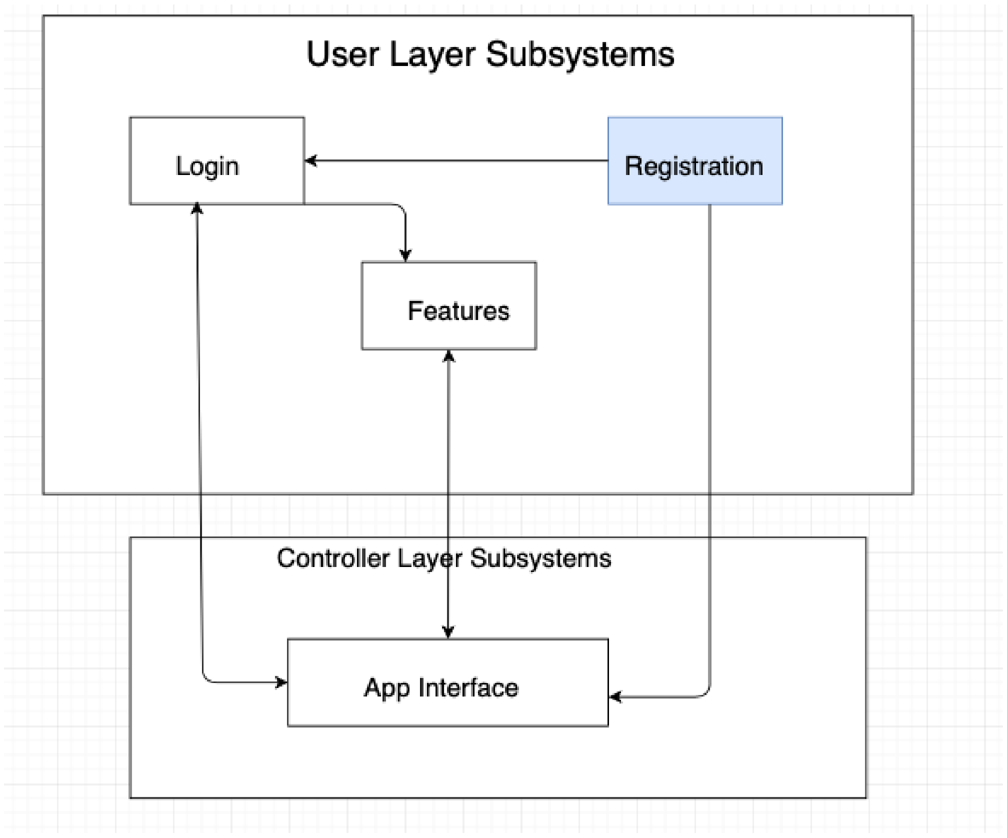

Figure 3: Registration subsystem overview

4.1.1 Assumptions

We assume that the user has an email and required credentials to create an account to move on to accessing the features of the application.

4.1.2 Responsibilities

The registration subsystem is responsible for creating a user account and adding it to the database which is linked to login subsystem and api interface for further step.

4.1.3 Subsystem Interfaces

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	The Registration subsystem interacts with the other subsystems providing the user credentials to register into the system database.	user name, email, password	registered or denied

4.2 LOG IN

Login being the Second subsystem for User Layer; is one of the steps towards being involved in the application. This subsystem asks user for their credentials to verify that the user exists in the system database. This system will be available in the home page along with the option to register an account.When the system verifies that the account exists, the user will be able to access application features.

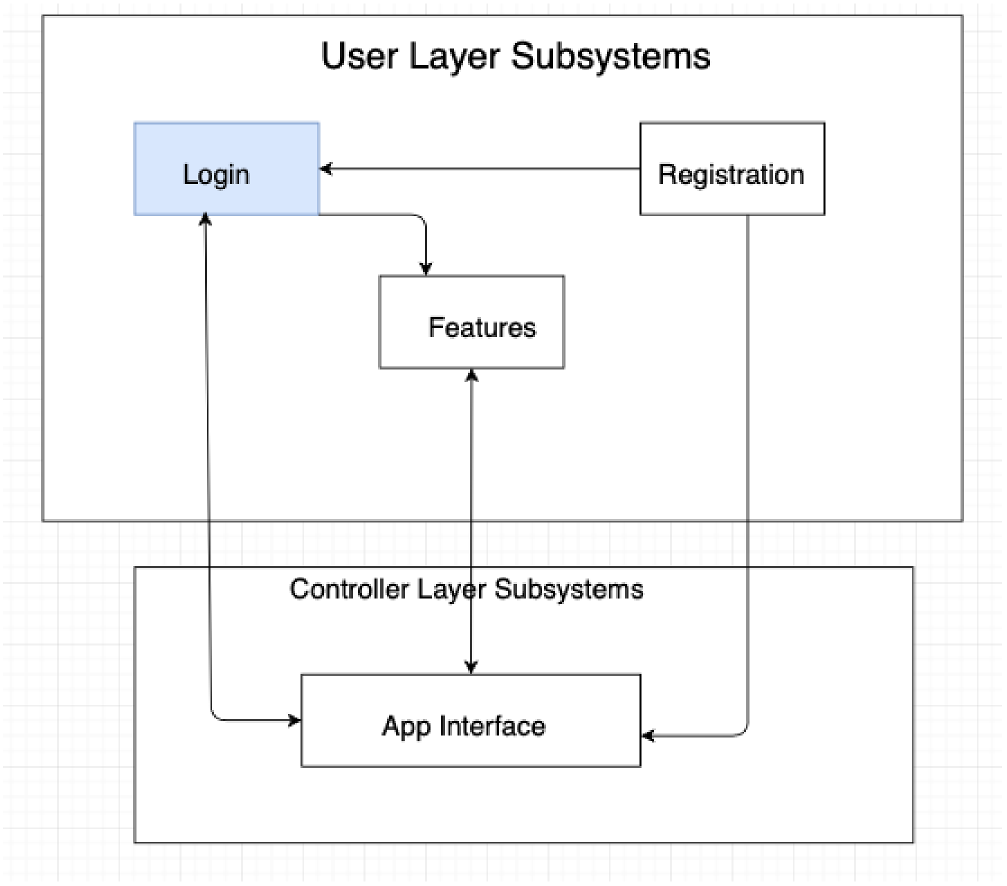

Figure 4: Log in subsystem overview

4.2.1 Assumptions

We assume that the user has a registered account in the system database to login and access personal dashboard and application features.

4.2.2 Responsibilities

The log in subsystem is responsible for allowing user to access the personal dashboard and account information. It is linked with providing application features to the user and API interface to check if the user exists in the database.

4.2.3 Subsystem Interfaces

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	The Login subsystem interacts with the Feature subsystem to provide the login credentials from the user to access the features.	user id, password	access or denial

4.3 FEATURES

Features of the application is another subsystem for User Layer that interacts with the API interface of the control layer to provide the dashboard or the page requested by the user up front.There is a two way interaction between the Feature subsystem of the User Layer and the app Interface of the Controller Layer.

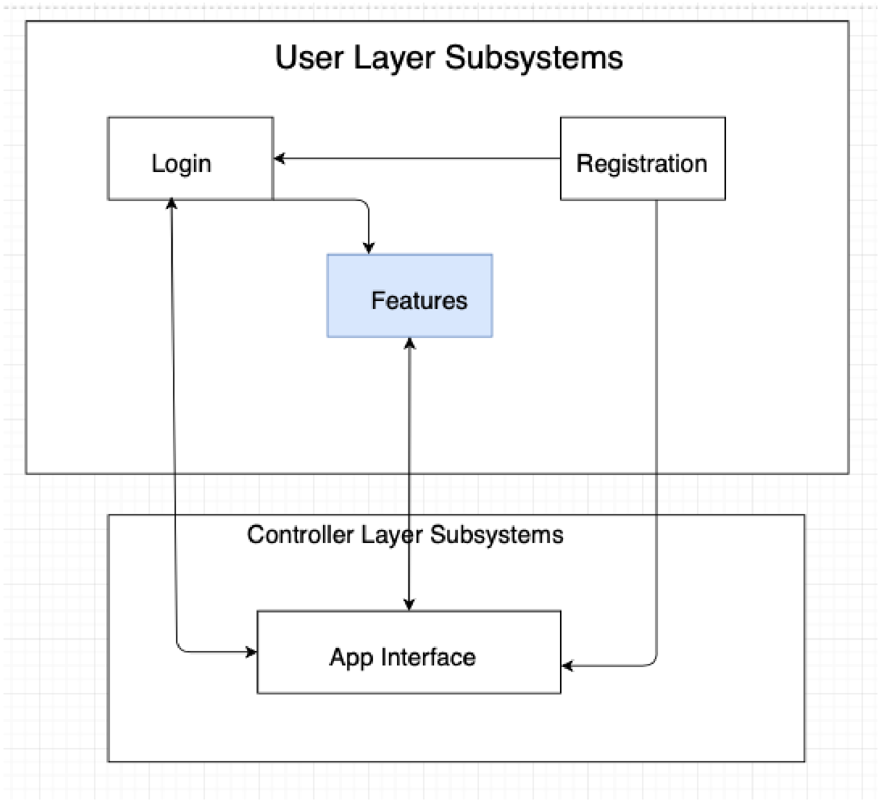

Figure 5: feature subsystem overview

4.3.1 Assumptions

We assume that the user has provided personal information to set up a dashboard with statistics and is able to get hold of the features of the application.

4.3.2 Responsibilities

The feature subsystem is responsible for providing logged in user with application features with personalized dashboard that has a two way interaction with the API interface.

4.3.3 Subsystem Interfaces

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	The Features subsystem interacts with the app interface to provide the system with login credentials form the user and get the user dashboard and statistics	data from django database via app interface	display features on gui

5 Controller Layer Subsystems

The main function of the controller layer is to act as a middle man between user layer and the api/server layer. Every request made by user layer must go through the controller layer to get data from database or any other web pages. It only consists of one subsystem called app interface subsystem.

5.1 App interface

This subsystem will get the data from user and pass to server side and vice-versa.

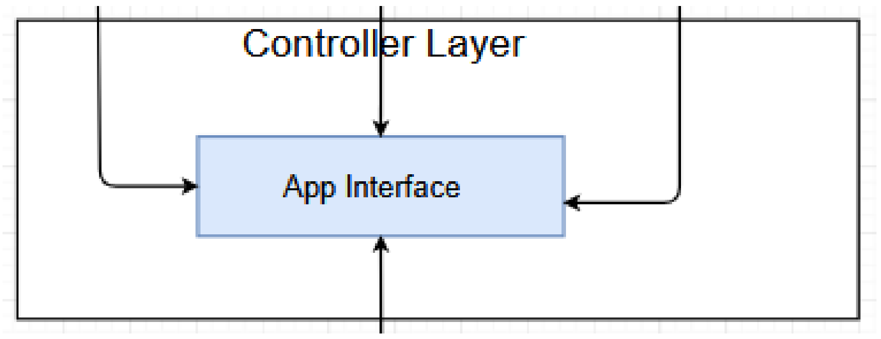

Figure 6: feature subsystem overview

5.1.1 Assumptions

Following are the assumptions for this subsystem:

- User has successfully registered for the account.
- User has added at least one bank account.
- User is trying to interact with different features.

5.1.2 Responsibilities

Each of the responsibilities/features/functions/services of the subsystem as identified in the architectural summary must be expanded to more detailed responsibilities. These responsibilities form the basis for the identification of the finer-grained responsibilities of the layer's internal subsystems. Clearly describe what each subsystem does.

5.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
#xx	Description of the interface/bus	input 1 input 2	output 1
#xx	Description of the interface/bus	N/A	output 1

6 API/Server Layer Subsystems

An application program interface (API) is a set of routines, protocols, and tools for building software applications. It specifies how software components should interact. It is meant to simplify communication protocol between a client and a server. The API layer connects the Controller layer and the Database layer

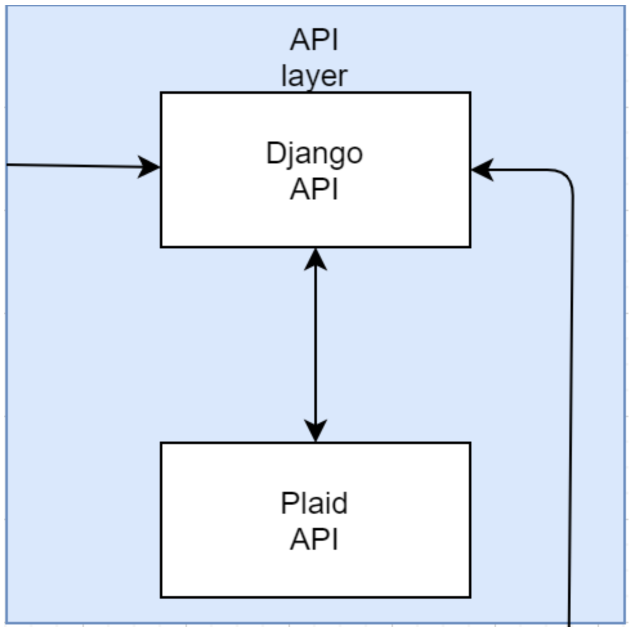

Figure 7: Backend section

6.1 Plaid API

The Plaid API allows developers to integrate transaction and account data from most major financial institutions into third party applications. The data includes merchant names, street addresses, geo-coordinates, categories, and other info. It focuses on using this data to interact with their bank accounts, check balances, and make payments.

6.1.1 Assumptions

We assume that the user has a bank account already. We assume that they have money in the account.

6.1.2 Responsibilities

Plaid enables third party applications to verify users' bank accounts in a way that doesn't share sensitive financial data

6.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

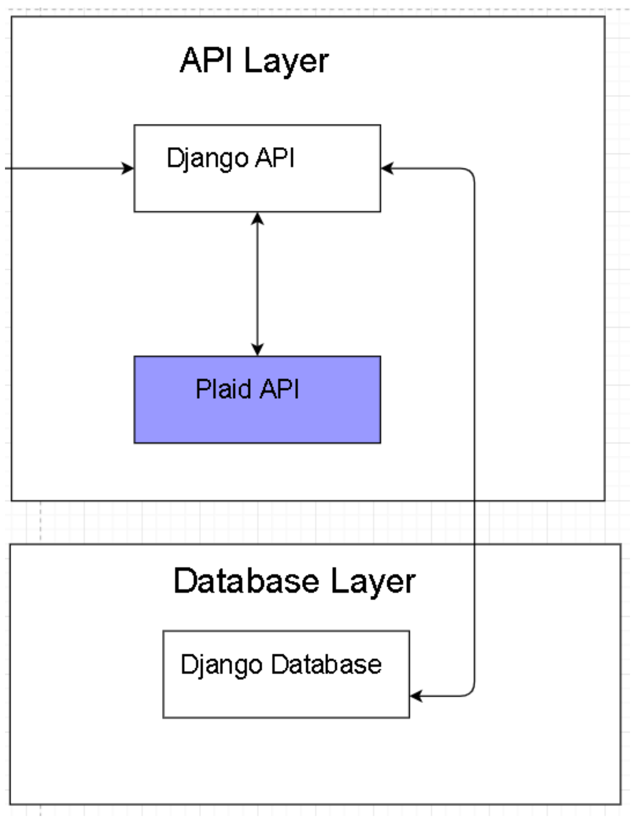

Figure 8: Plaid subsystem overview

Table 6: Subsystem interfaces Plaid

ID	Description	Inputs	Outputs
#1	Transaction name	Chicken Fried Rice Boba	Food
#xx	Cost	Cost 1	Total cost

6.2 Django API

Django is a web framework designed to help you build complex web applications simply and quickly. It will also help in communicating with the database.

6.2.1 Assumptions

We assume the user provides us with correct data for our database.

6.2.2 Responsibilities

HTTP response handling, content type negotiation using HTTP Accept headers.
subsubsectionSubsystem Interfaces

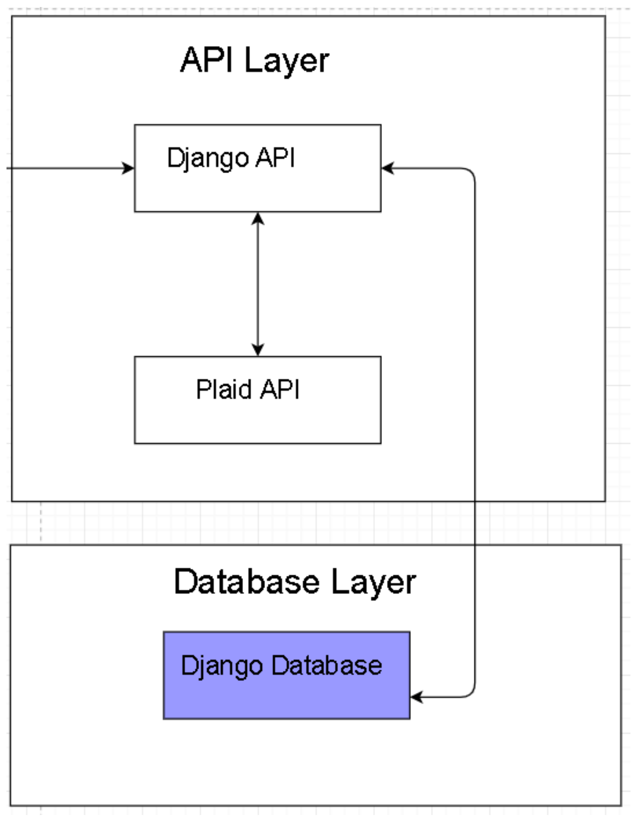

Figure 9: Django subsystem overview

7 Database Layer Subsystems

A database is used to store data. The database layer will provide persistent and secure storage of all entities of the application. The database layer will communicate with the API layer.

7.1 Django Database

The Django Database receives requests from the API layer and stores or retrieves user credentials, plaid tokens, user settings, and other data.

7.1.1 Assumptions

We assume the Django API is properly configured to send valid requests to the database.

7.1.2 Responsibilities

The responsibilities of the Database Layer are persistent and secure data storage of the application's entities.

7.1.3 Subsystem Interfaces

The database layer has two main types of interfaces: storage and retrieval.

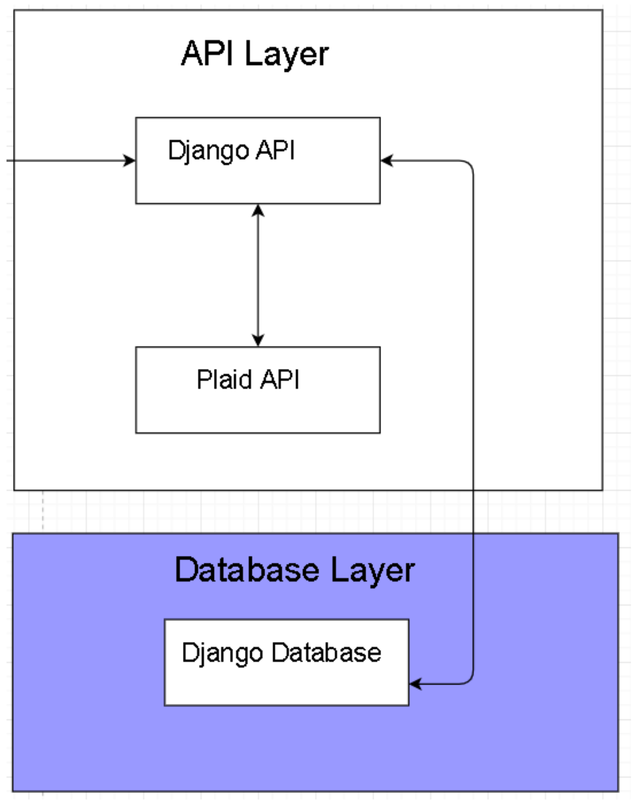

Figure 10: Database subsystem overview

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#01	Storage Request	User Data	N/A
#02	Retrieval Request	N/A	User Data