# Senior Design Final Report

## Drawdown Interface (DDI)

<table>
<thead>
<tr>
<th>All Account Balances</th>
<th>Balances by Plan</th>
<th>Balances by Contact</th>
<th>Flagged Accounts (Nonmatching)</th>
<th>Home</th>
<th>Analysis Tool</th>
</tr>
</thead>
</table>

**Drawdown Interface [ICUP]**

Cal State LA ECST Senior Design Project - Sponsored by LA County, Regional Planning

**Students:** Pranil Dahal, Iovanni Enriquez, Wesley Hwang, Cesar Rojas, Vay Tang

**Advisor:** Yuqing Zhu

**Liaisons:** Tony Alcaraz

---

**Version 1.0 - 05/10/2019**

**Team Members:**
- Pranil Dahal
- Iovanni Enriquez
- Wesley Hwang
- Cesar Rojas
- Vay Tang

**Faculty Advisor:**
- Dr. Yuqing Zhu

**Liaisons:**
- Tony Alcaraz
# Table of Contents

1. Introduction 2
   1.1. Background 2
   1.2. Design Principles 2
   1.3. Design Benefits 3
   1.4. Achievements 3
2. System Architecture 3
   2.1. Overview 3
   2.2. Data Flow 4
   2.3. Implementation 5
3. Conclusions 6
   3.1. Results 6
   3.2. Future 6
4. References 7
1. Introduction:

1.1. Background:

Regional Planning started in Los Angeles County in 1922 with the establishment of the Regional Planning Commission (RPC), a body that still exists and is responsible for advising the Board of Supervisors on all planning matters. The Department of Regional Planning (DRP) became an independent department in 1974 and continues to support the Commission and Board of Supervisors.

To fulfill its responsibilities the RPC formulates policies and conducts regular public hearings weekly. Tentative subdivision tracts, zoning changes, variances and permits, countywide and community plan policies are presented and acted upon the Commission. DRP is responsible for both providing immediate responses to enforcement and permitting requests, and considering long-range development issues. Therefore, DRP must consider a broad range of constituents, from individual homeowners, to large landowners, to community groups, to future generations of Los Angeles County residents.

As Los Angeles County continues to grow, the workflow for employees of the DRP will increase. This means more data to process and review. There needs to be some way to streamline the workflow so DRP can keep up with the demands of its constituents. Therefore, DRP has teamed up with California State University, Los Angeles to develop a web application to display Conditional User Permit (CUP) data in an efficient, reliable, and simple way. This web application is called Drawdown Interface (DDI).

1.2. Design Principles:

The main goal of DDI is to centralize all data relating to CUP Deposit Accounts and display the data in a user-friendly interface. The application needs to be simple and it needs to not make excessive calls to the LA County database, so that the application does not hog the database’s traffic. Also since there is no current interface that allows the employees of DRP to easily access and view the data, the application needs to be simple in design so that implementation of the application can be completed within the allotted time frame. The application will only allow the user to read and not edit or delete the data. DDI will also allow for custom searches and allow the user to drill into an instance of data for more detail.
1.3. Design Benefits:

By having a simple design in mind, the user will be able to easily navigate through DDI and search for the desired information to complete day-to-day tasks in the DRP. Our application also allows for scalability so that the application will have full functionality as the LA County database expands.

1.4. Achievements:

Over the course of the academic year, our team has been able to develop a functional application that provides access to data from a local database and displays it in a simple but aesthetically pleasing fashion. Using Apache Maven, the build process for our application has been simplified and allowed DDI to be easily maintained and expanded for future modifications.

2. System architecture:

2.1. Overview:

The architecture for DDI can be broken down into two main factors: Frontend and the Backend

Here is a diagram (DFD level 0) that shows how this architecture works at a low level:

- **Frontend**: Interface that consumes REST APIs provided by the Backend and displays the data in a web UI
• **Backend**: Queries the database to retrieve financial and personal data about different CUP accounts

### 2.2. Data Flow:

Here is an overview of DDI and how the data flows through the application:

There are three major modules in this system. Here is a brief overview of them:

#### 2.2.1. Data Access Module (DAM): It will create queries based on the data received from the HRHM, which will then return a result set from the database.
2.2.2. **HTTP Request Handle Module (HRHM):** The HRHM will be the first module to see data from the Frontend and will connect the backend with the Frontend. It will receive HTTP requests from the front end and write out queries based on the data that is requested. After receiving JSON data from the DAM, the data will be forwards back to the Frontend.

2.2.3. **Parameter Processing Module (PPM):** The central processing module for the Frontend. PPM receives the parameters from the UI (example: account ID, plan ID, date ranges, etc) and constructs the API query. The API query gets sent to the HRHM module (2.1.2). PPM also receives the JSON data and renders the necessary UI components and passes them on to the UI.

### 2.3. Implementation:

To display the following reports:

1. All Balance Reports.
2. Individual Balance by Contract.

The project was split into two sections to allow for efficient development: Data Extraction and User Display. Each section plays a key role in presenting the progression of the project.

2.3.1. **Data Extraction:** The data extraction mechanism has been split into three major section on the backend. Every row of SQL data that gets extracted and displayed to the user goes through the following steps.

   I. Rowmapper Java Object
   II. Extraction Class
   III. Controller Class

First, the Extraction Class used pre-build SQL Queries to extract rows of data from the database.

After that, each row get mapped to a java Object that corresponds to the type of data that is being extracted from the database. For example, NonMatchingAccount.java class is used to create instances of Java Objects that hold the information about all the accounts with non-matching account numbers.

Finally, the Controller Class provides the endpoint that will be used by the frontend, and the data gets supplied in JSON format.
2.3.2. **User Display:** Using Angular components, a component controls a patch of screen, we can refer to as a view. For example, individual components define and control each of the following:

1. Home
2. All Account Balances
3. Balance by Contact
   a. Transaction History
4. Balance by Plan
5. Non-Matching Accounts

I. Home Component
   - Provides user with access to each of the reports.

II. All Account Balances Component
   - Displays a full list of all account balances available from the database

III. Balance by Contact Component
   - Displays the contact the user is searching for.

IV. Balance by Plan Component
   - Displays the plan the user is searching for.

V. Non-Matching Component
   - Displays all non-matching accounts to the user.

3. **Conclusions:**

3.1. **Results:**

We have created classes and methods arranged in a structure that is easy to follow and allows for further development.

Using current and popular frameworks and technologies, we have created a web application that retrieves data from a local database and designed an aesthetically pleasing user interface.

3.2. **Future:**

Implementation of a data analysis tool. Since there is a lot of data to be worked with, there may be important information that can be extracted from the plethora of data the LA County processes.
4. References:

N/A