**Advertiser Analytics**

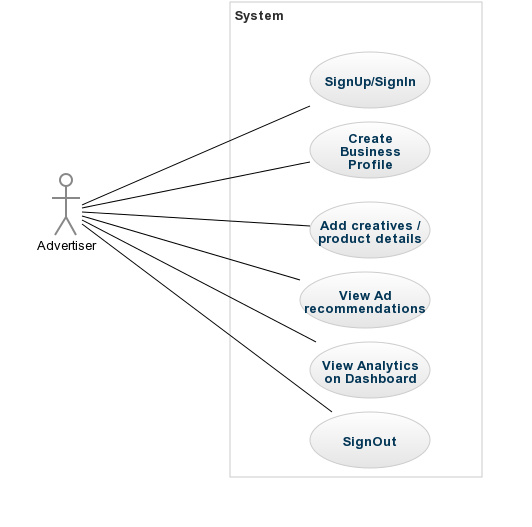
|  |
| --- |
| Project Workbook |
|  |

|  |
| --- |
| By  Sai Sudha Kiran Reddy Dwaram  Naga Santosh Sairam sibyala  Srinivasa Joga Commuri  Venkat kotu |
|  |
| 10/27/15 |
| **Advisor:** Prof Chandrashekar Vuppalapati |

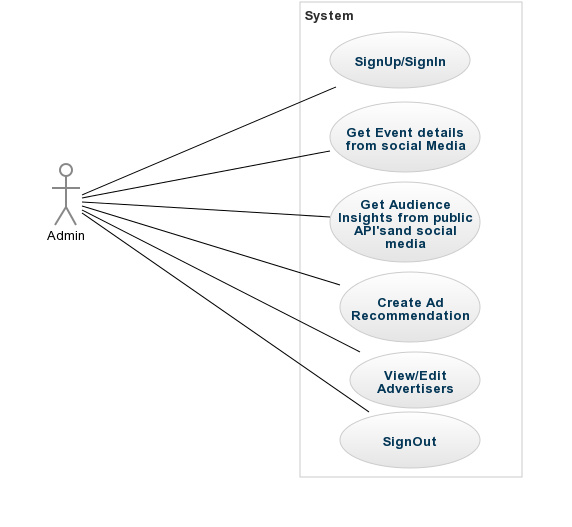
# Project Design

We have created use case diagram to represent the use cases for end users, ERD diagram and relational schema to describe our application entity relational model, sequence diagram to describe the flow of application and class diagram which shows the entities and their properties and methods.

Use case Diagram 1:



Use case Diagram 2:



Description:

|  |  |
| --- | --- |
| Use Case ID | 1 |
| Use Case Name | User Signup |
| Use Case Description | Sign up of a user for the first time |
| Actor | Advertiser, Admin |
| Pre-Conditions | The Advertiser or Admin should have valid email id. |
| Basic Flow | 1. Advertiser or Admin enters the details that are required. 2. The entered details are validated against the required format. 3. If the validation of entered details is successfully done then the details are stored in the database. 4. Else Go To Step 1 |
| Post-Conditions | Advertiser or Admin details should be successfully stored in database. |

|  |  |
| --- | --- |
| Use Case ID | 2 |
| Use Case Name | User Sign In |
| Use Case Description | The user signing in to the system |
| Actor | Advertiser, Admin |
| Pre-Conditions | The Advertiser should Sign up. |
| Basic Flow | 1. Advertiser or Admin enters the email id and password. 2. The entered details are validated against the details that are already stored in database. 3. If the validation of entered details is successfully done then the advertiser is redirected into home page of the system. 4. Else Go To Step 1 |
| Post-Conditions | Advertiser or Admin should successfully enter homepage of system. |

|  |  |
| --- | --- |
| Use Case ID | 3 |
| Use Case Name | Create Business Profile |
| Use Case Description | Creating Business Profile for User |
| Actor | Advertiser |
| Pre-Conditions | The Advertiser should have logged in to create business profile. |
| Basic Flow | 1. Advertiser enters the business profile details that are required. 2. The entered details are validated against the required format. 3. If the validation of entered details is successfully done then the details are stored in the database. 4. Else Go To Step 1 |
| Post-Conditions | The business profile details of Advertiser should be successfully stored in database. |
| Use Case ID | 4 |
| Use Case Name | Provide Creative/Ad details |
| Use Case Description | The details of creatives or ads should be provided |
| Actor | Advertiser |
| Pre-Conditions | The Advertiser should be logged in to provide details about creatives |
| Basic Flow | 1. Advertiser enters the details about the creatives that are required like adSize, media\_type etc… 2. The entered details are validated against the required format. 3. If the validation of entered details is successfully done then the details are stored in the database. 4. Else Go To Step 1 |
| Post-Conditions | The provided creative details should be successfully stored in database. |

|  |  |
| --- | --- |
| Use Case ID | 5 |
| Use Case Name | View Suggestions |
| Use Case Description | The advertiser views the suggestions provided by the system about creatives. |
| Actor | Advertiser |
| Pre-Conditions | The Advertiser should be logged in and should have provided details about creatives. |
| Basic Flow | 1. The system takes the details of creative. 2. It searches the database table for suggestions about creative. 3. If there are any suggestions then those are returned to advertiser. 4. Else No suggestions are provided. |
| Post-Conditions | The suggestions about creative should be returned to Advertiser. |

|  |  |
| --- | --- |
| Use Case ID | 6 |
| Use Case Name | Use dashboard |
| Use Case Description | Using a dashboard to view analytics |
| Actor | Advertiser, Admin |
| Pre-Conditions | The Advertiser or admin should be logged in successfully to use dashboard. |
| Basic Flow |  |
| Post-Conditions | Advertiser or Admin should be able to use dashboard for the purpose of analytics. |

|  |  |
| --- | --- |
| Use Case ID | 7 |
| Use Case Name | Sign out |
| Use Case Description | Sign out of a user from the system after using it. |
| Actor | Advertiser, Admin |
| Pre-Conditions | The Advertiser or Admin should be logged into the system. |
| Basic Flow | 1. Advertiser or Admin presses sign out button. 2. The actor’s session is destroyed. 3. The actor is redirected to login page. |
| Post-Conditions | Advertiser or Admin should be redirected to login page. |

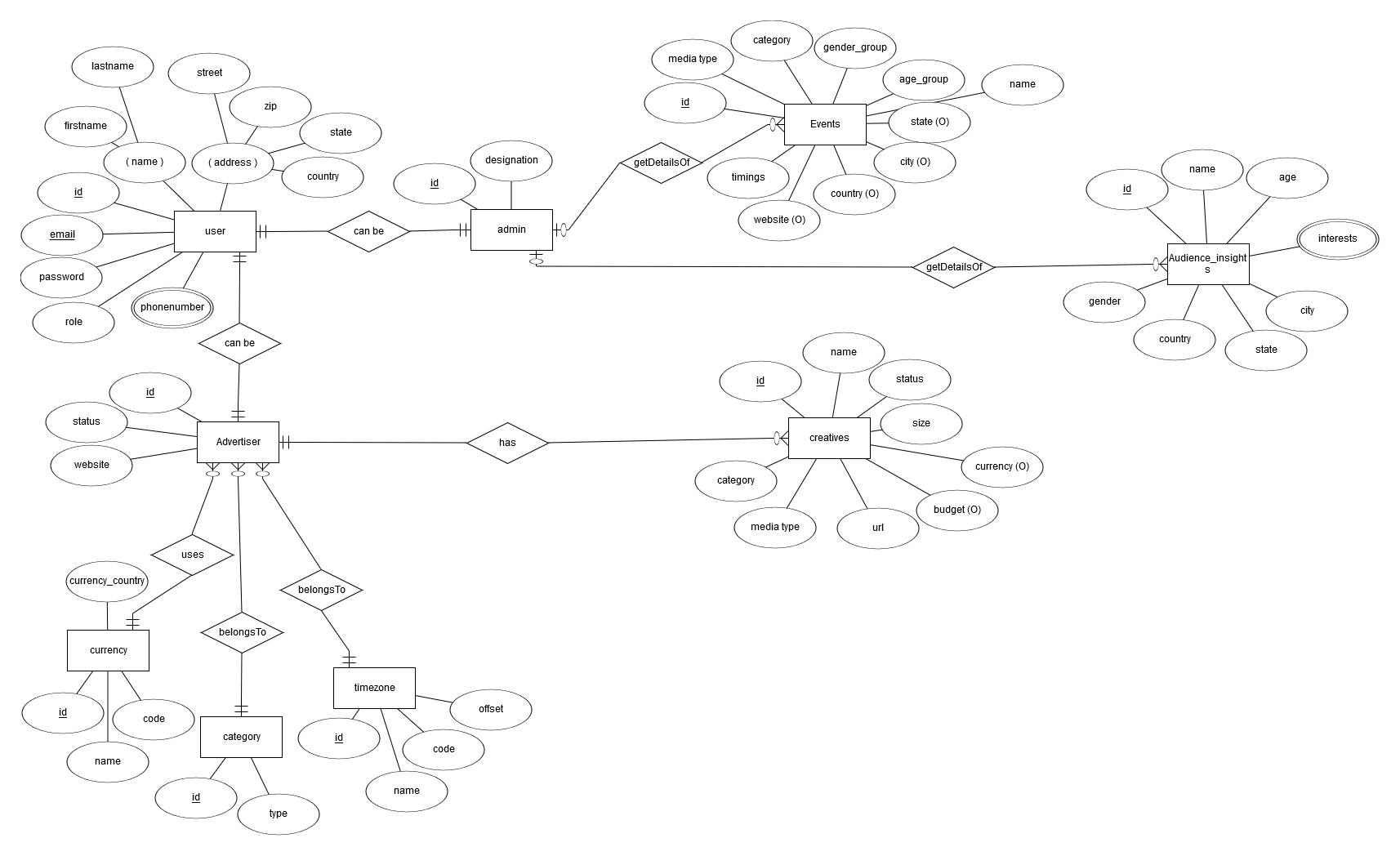
|  |  |
| --- | --- |
| Use Case ID | 8 |
| Use Case Name | View Advertisers |
| Use Case Description | Viewing the advertisers who ask for recommendations. |
| Actor | Admin |
| Pre-Conditions | Admin should be logged in to view advertisers |
| Basic Flow | 1. Admin should press “View Advertisers” button. 2. Then a database select operation is done. 3. The results of the database operation are returned to admin. |
| Post-Conditions | Admin should be able to view advertisers. |

|  |  |
| --- | --- |
| Use Case ID | 9 |
| Use Case Name | Edit Advertisers |
| Use Case Description | Edit the details of Advertisers if needed. |
| Actor | Admin |
| Pre-Conditions | The Admin should be logged in and the advertiser details to be changed should be provided. |
| Basic Flow | 1. Admin should press “Edit Advertisers” button which will redirect to page to change details. 2. The entered details should be updated in the database table. 3. If they are successfully updated in database then details are edited. 4. Else Go To Step 1 |
| Post-Conditions | Advertiser details should be updated in the database table. |

|  |  |
| --- | --- |
| Use Case ID | 10 |
| Use Case Name | Search Events |
| Use Case Description | Searching the details of Events from Social Media based on advertisers creatives. |
| Actor | Admin. |
| Pre-Conditions | The Admin should be logged in and details of creatives should be provided. |
| Basic Flow | 1. The details about the creatives are taken by admin. 2. Using those details the search operation is performed on database table which contain information about events that are taken from social media like web, TV, mobile etc… 3. The results of above operation are returned to the admin. |
| Post-Conditions | The event details should be returned to admin. |

|  |  |
| --- | --- |
| Use Case ID | 11 |
| Use Case Name | Search Audience Insights |
| Use Case Description | Search the details of audience insights based on advertisers information |
| Actor | Admin |
| Pre-Conditions | The Admin should be logged in and should have information about creatives. |
| Basic Flow | 1. The details about the creatives are taken by admin. 2. Using those details the search operation is performed on database table which contain information about audience insights that are taken from social media like web, TV, mobile etc… 3. The results of above operation are returned to the admin. |
| Post-Conditions | The audience insights should be returned to admin. |

## ERD Diagram

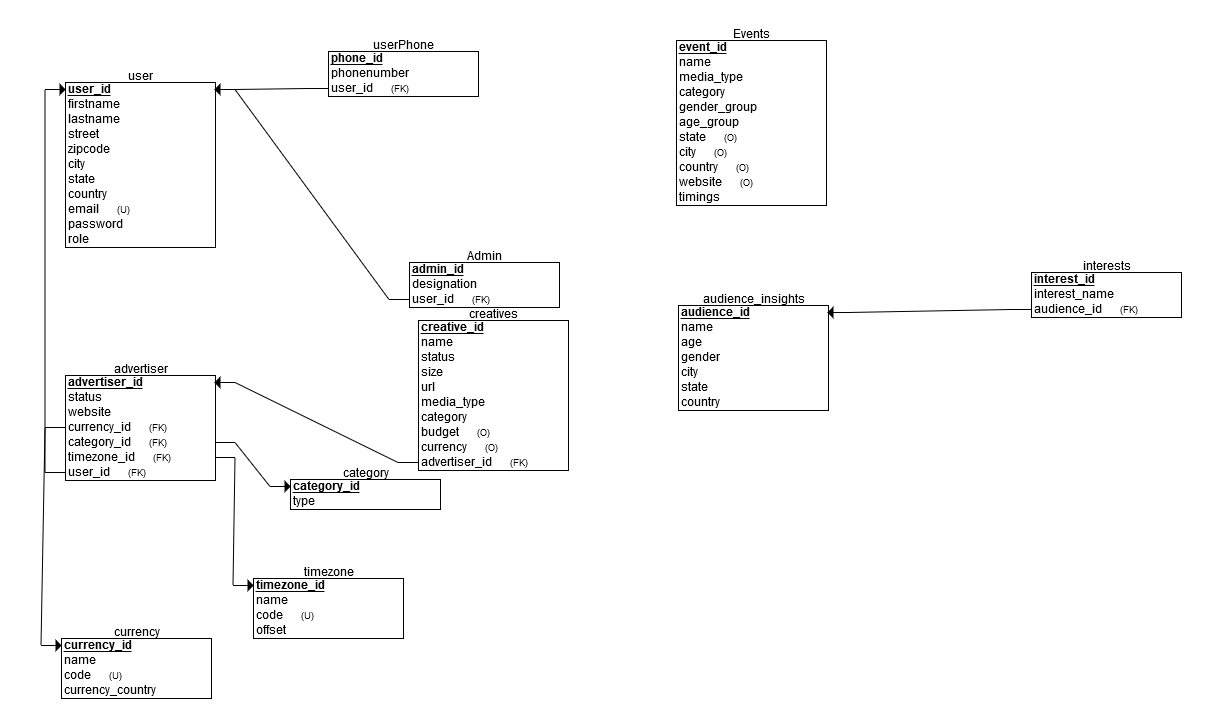


Description

Our project contains 9 core entities, below is the description of the each entity in detail.

* ***User Entity:*** This is the user entity, which contains all the user details. This entity has a unique id and email attribute with which a user can be uniquely identified throughout the system. It also includes various multivalued (like phone number) and composite attribute like name attribute which is composed of first name and last\_name, address attribute which is composed of state, zip, city and country properties.
* ***Advertiser Entity*:** Each user can be an advertiser; each advertiser consists of all the user attributes along with the advertiser specific attributes like Status, company name, company website.
* ***Currency Entity:*** Advertiser uses this entity to specify his time zone preference. It has the unique currency\_code attribute, currency country and Name.
* ***Category Entity****:* Each Advertiser has a specific category like (social, science & technology, kids, entertainment etc.) This information is maintained using this Category Entity. This has unique id and the category name.
* ***Admin Entity:*** Each user can be an admin. This has certain Role and authorization specific details of each user.
* ***Events Entity:*** This is one of the important entities in our ERD. This entity contains all the events details that are happening. System gets the information from about the events based on the advertiser inputs using this entity. This has the following attributes, unique id, Event name, Time, category of the event, media\_type (like web or in person event etc.), gender\_group which specifies for which gender this event is for, age\_group which specifies which age group is this event for, web\_site information, location details (City, state, zip, country).
* ***Creative Entity:*** This entity is used to store all the creative/ad specific details of the advertisers. This has unique id, creative name, Status which specifies if this is still active or not, media\_type of the creative (like video, audio or native ad), size of the creative, url which is a optional attribute.
* ***Audience Insights:*** This entity contains all the details about the users insights about a particular product or thing. This has unique id, name, age of the user, gender, country, state, city and a multivalued interests of the user. This entity is used to display insights to the advertiser based on his input details.

Relational Schema



Description:

The Advertiser table and Admin table has foreign key references to User table, which indicates a user can be an advertiser or Admin.

Creative and Advertisers are have a one to many relation ship, means each Advertiser can have many creative under his account, so Creative has a foreign key reference to the advertiser table.

Advertiser and Category are related to each other, Advertiser has the foreign key references to the category table.

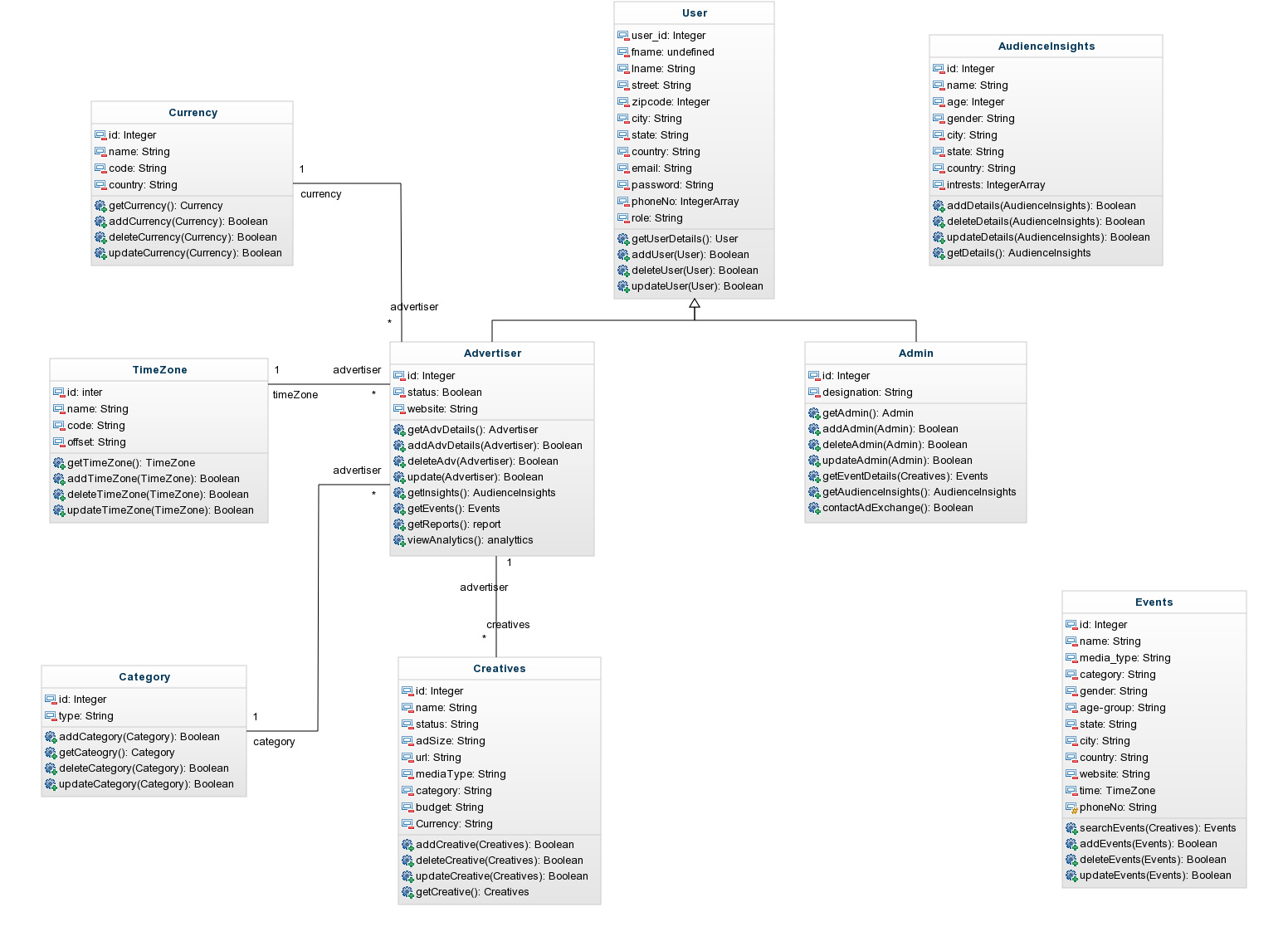
Each Advertiser has a Currency so these two entities are related to each other so advertiser has a foreign key reference to the Currency Table.

Each Advertiser has a TimeZoone preference so these two entities are related to each other so advertiser has a foreign key reference to the Time Zone Table.

Events table contains all the information about Events. It is data table just used to get information and is not related to other tables.

Each audience has many interests so we have one to many relation between the audience and interests table, so Interests has foreign key reference to the audience table.

Class Diagram



Description:

***User***: This class has the private properties of the user like user\_id, email, name, firstname, lastname, street, city, zipcode, country, password hash, phoneno, role. This has four public methods getUserdetails(), addUser(user), deleteUser(user), updateUser(user) which perform CRUD operations on user object.

***Advertiser****:* This class has the private properties id, status, website, companyname. The following are the public methods on this class which are used to perform various crud operations getAdvDetails(), addAdvDetails(), deleteAdvDetails(), updateAdv() and other operations like getInsights() which is used to get the audience insights details, getEvents() which gets the events from Events table, getReports() to get the reports, viewAnalytics() which perform necessary opeartions to display the analytics based on the advertiser input.

***Events****:* This class has the private properties id, event name, media type of the event,

Even fall under which category, gender preference of the event, age-group of the event, location and timezone details. It has four public methods searchEvents(), addEvents(), deleteEvents(), updateEvents() to perform crud operations on the Event object.

***AduienceInsights****:* It has the private properties about the audiences and their interests. This has id, name, age, gender, location and the various interests about the user. It has four public methods addDetails(), updateDetails(), deleteDetails(), getDetails() which perform the crud operations.

***Creative*:** This class has the following private properties creative\_id, creative\_name, creative\_status, ad\_size, url, media\_type of the creative, creative\_category. It has four public methods to perform the crud operations on the creative.

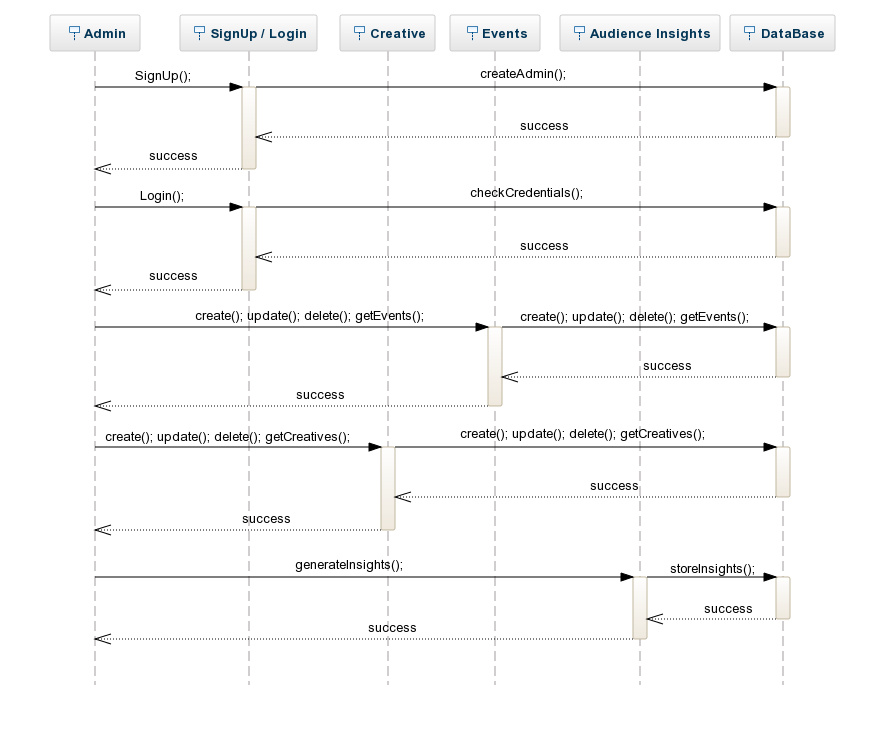
***Admin****:* It has the private properties id, designation, authorization related to admin. This has the public methods getAdmin(), updateAdmin(), deleteAdmin(), addAdmin() to perform the crud operations. Apart from this it has getEventDetails() , getInsights() methods to retrieve the event and insights details.

***Category****:* This class two private properties id and category\_type and four pubic methods addCategory(), updateCategory(), deleteCategory() and getCategory() to perform the crud operations on the category object.

***TimeZone****:* This class has four private properties id, name, code, offset and four public methods getTimezone(), addTimeZone(), deleteTimeZone(), updateTimeZone() to perform operations on this object.

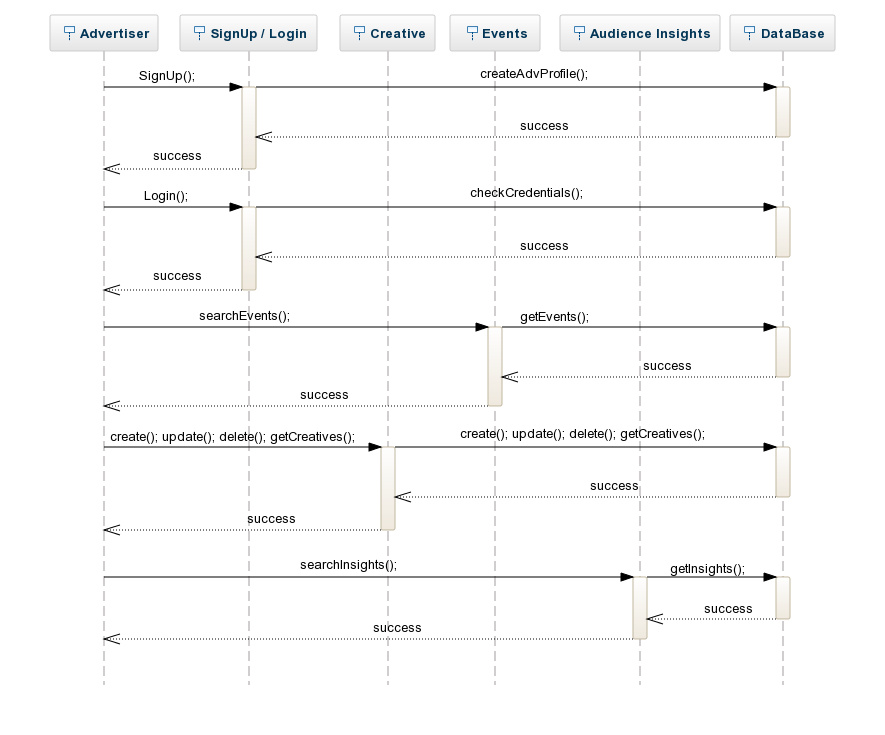
***Currency****:* This class has four private properties id, name, code country and four public methods getCurrency(), addCurrency(), updateCurrency(), deleteCurrency() to perform the curd operations.

Sequence Diagram



Description:

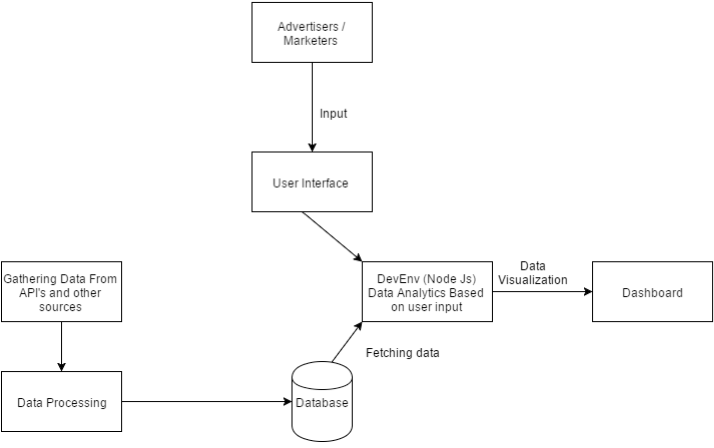
The sequence diagram describes the Admin functionality flow of logic. The Admin<<actor>> will have to create a profile using the signup page in the UI, then the profile details will be entered into the database. The Admin will have to login into the account using the credentials in the login page, then the credentials will be verified from the database in the backend. The Admin will have various functionalities in the Events module<<UI>> like creating events, adding new events, delete and get the list of various events all these CRUD operations will update the data base in the backend accordingly based on the user input from the UI. Similarly, the Admin can perform various operations on the Creatives module<<UI>>. The Admin uses the data collected from various API’s and generates audience insights for various products and stores the insights in the database in the backend.



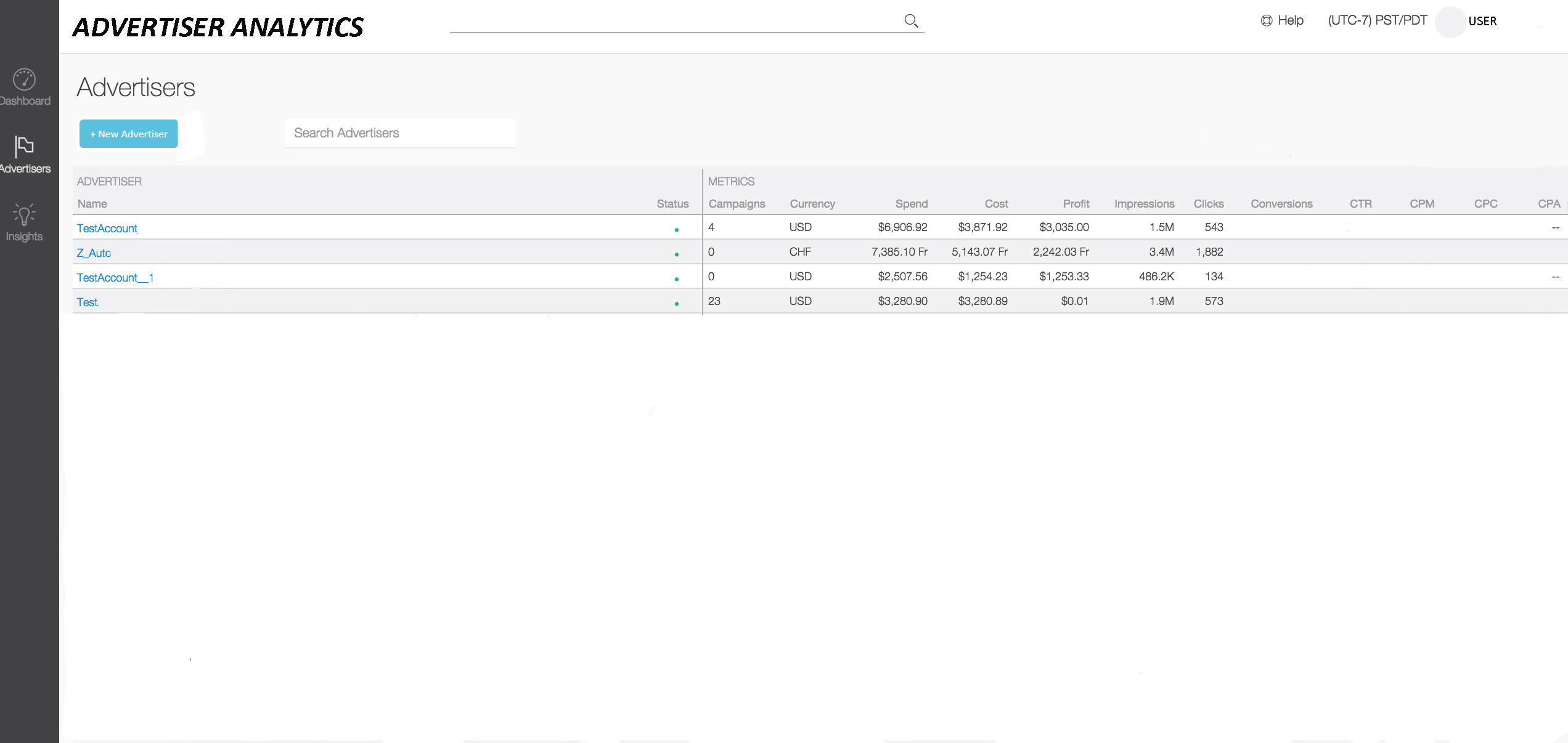
Description

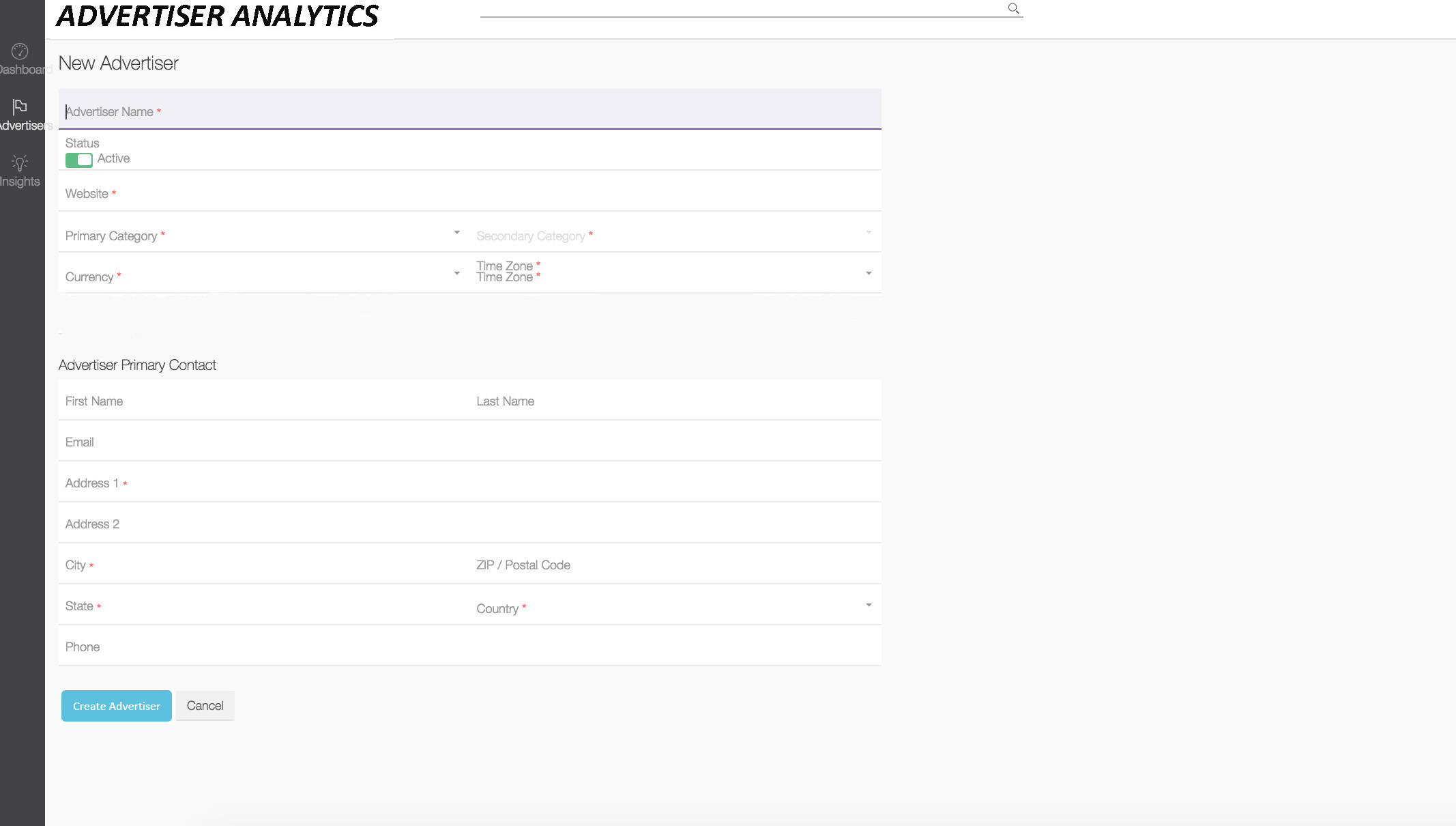
The sequence diagram describes the Advertiser functionality flow of logic. The Advertiser<<actor>> will have to create a profile using the signup page in the UI, then the profile details will be entered into the database. The Advertiser will have to login into the account using the credentials in the login page, then the credentials will be verified from the database in the backend. The advertiser can search for Events in the Events Module in the UI, then the backend logic fetches the results from the database based on the user input. Similarly, the Advertiser can search for audience Insights in the audience Insights Module in the UI and the backend logic fetches the results from the database and displays them on the UI. the Advertiser can perform operations on the Creatives module like create update delete and get creative details for a previous creative or ads, the user input from the UI will be updated in the database accordingly.

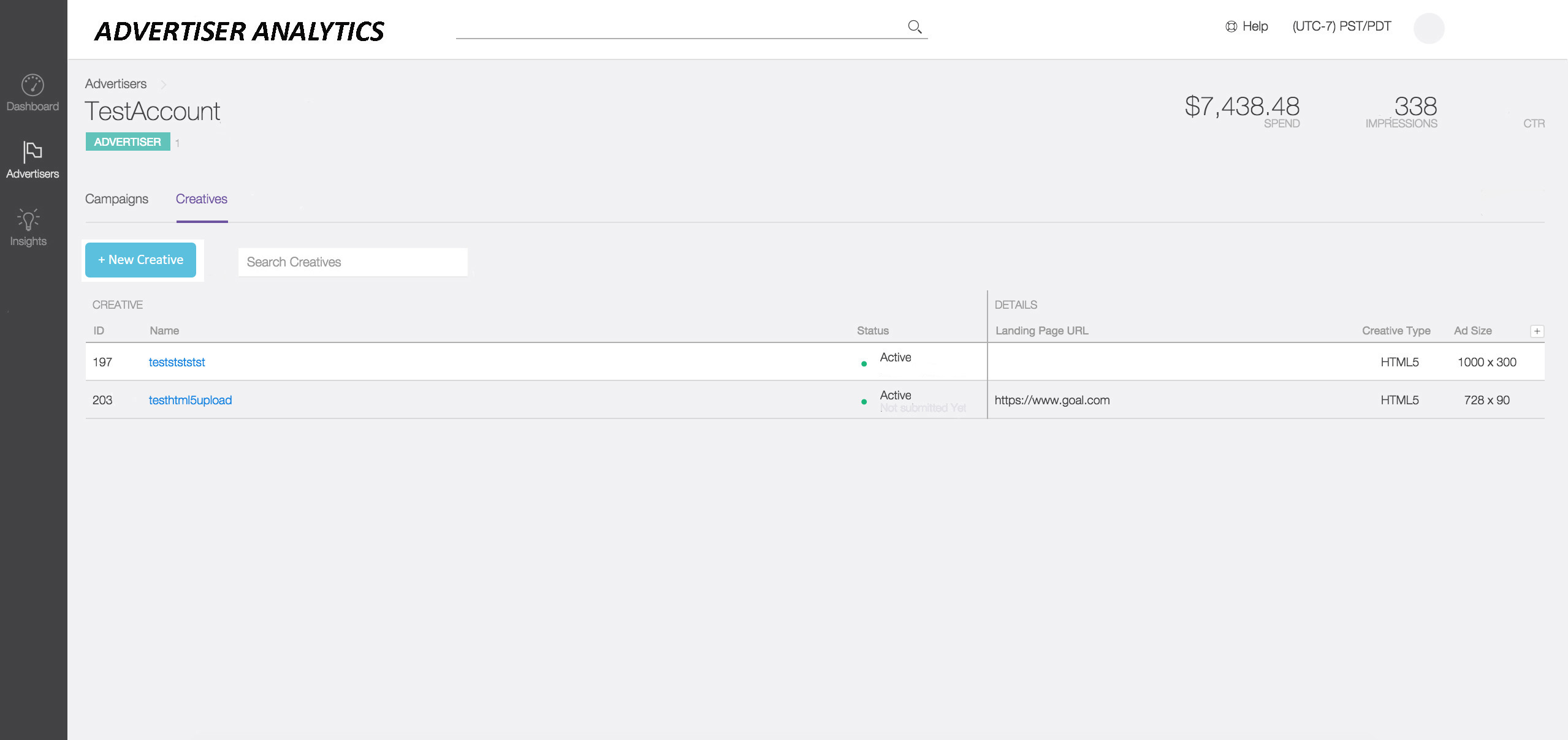
**Data Flow Diagram:**

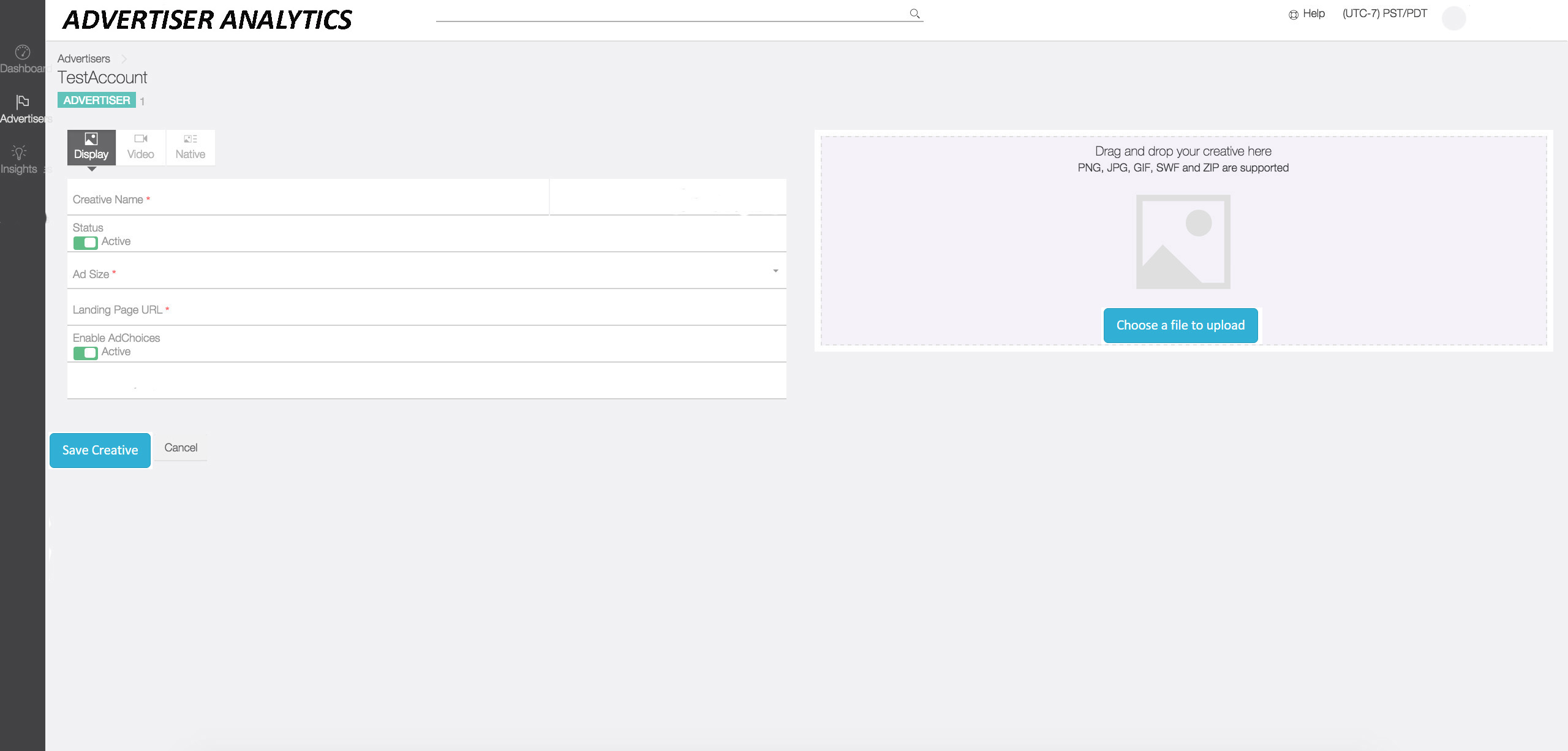


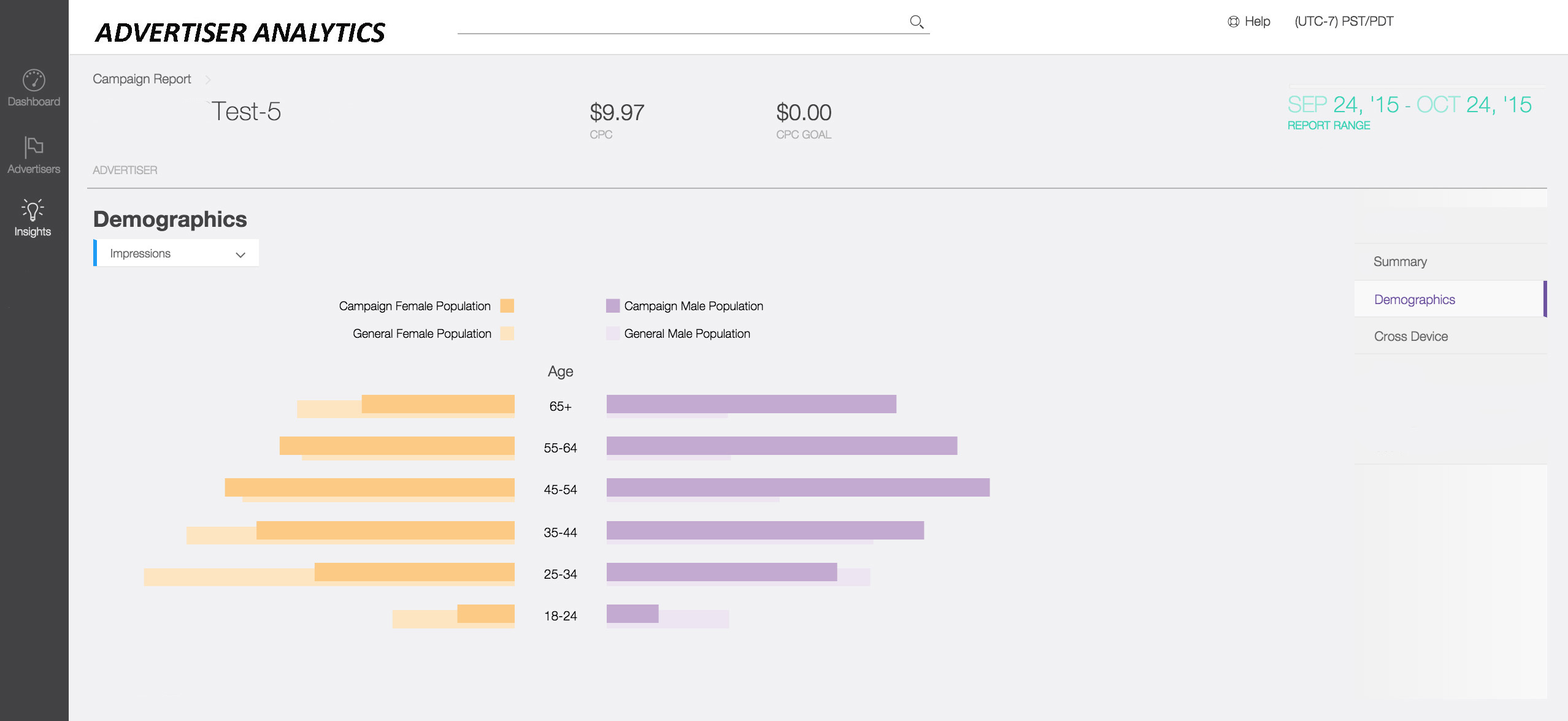
# Mockups

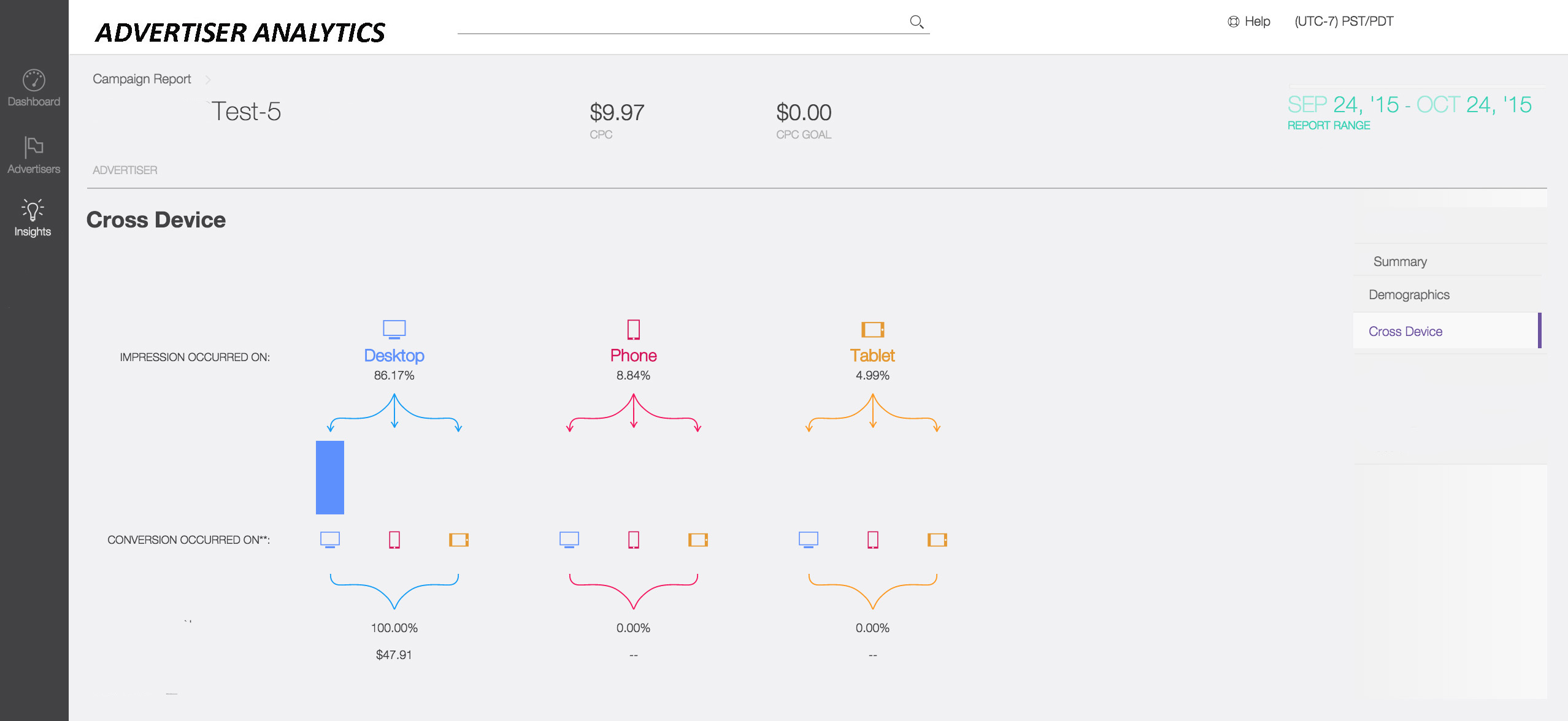


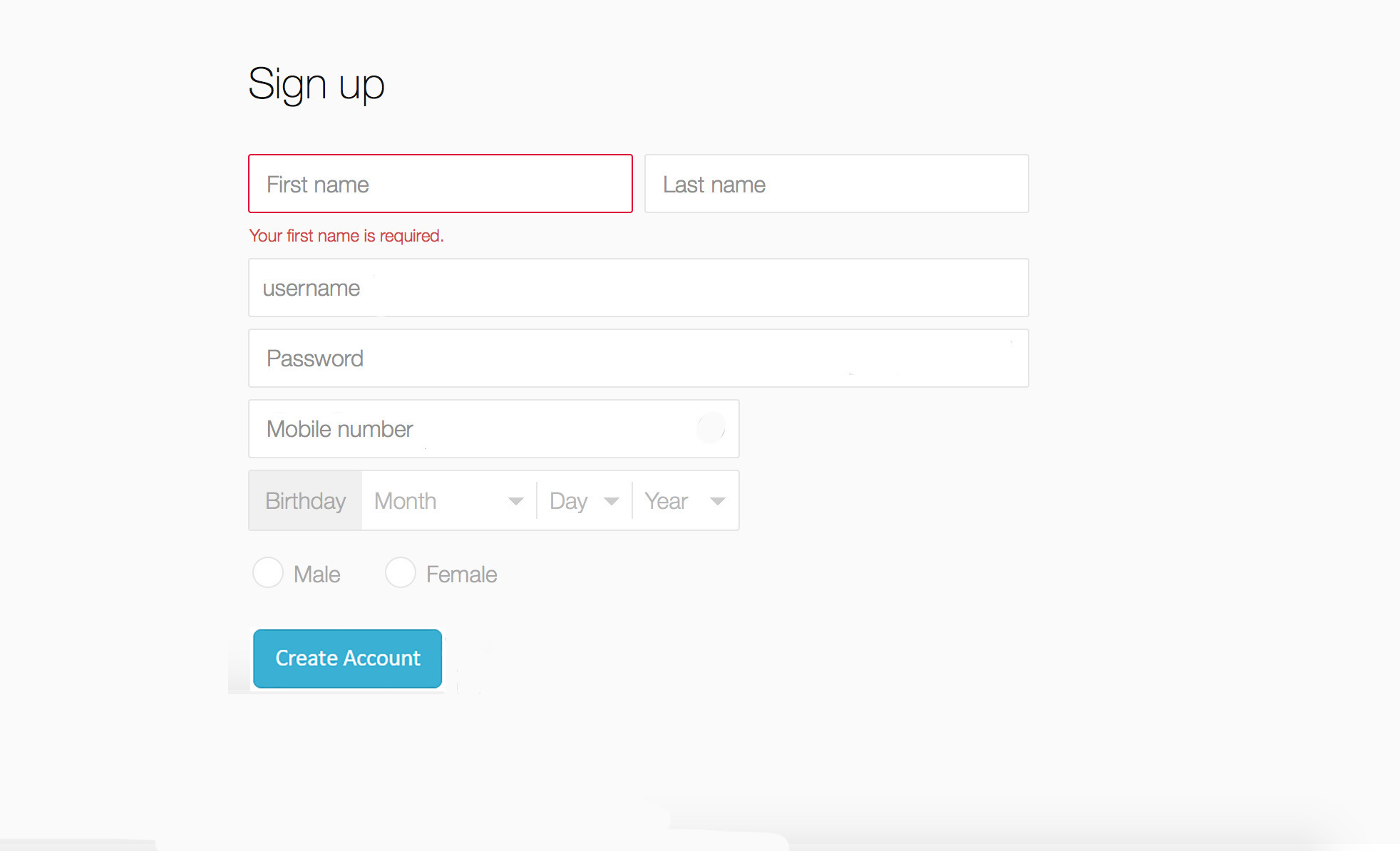


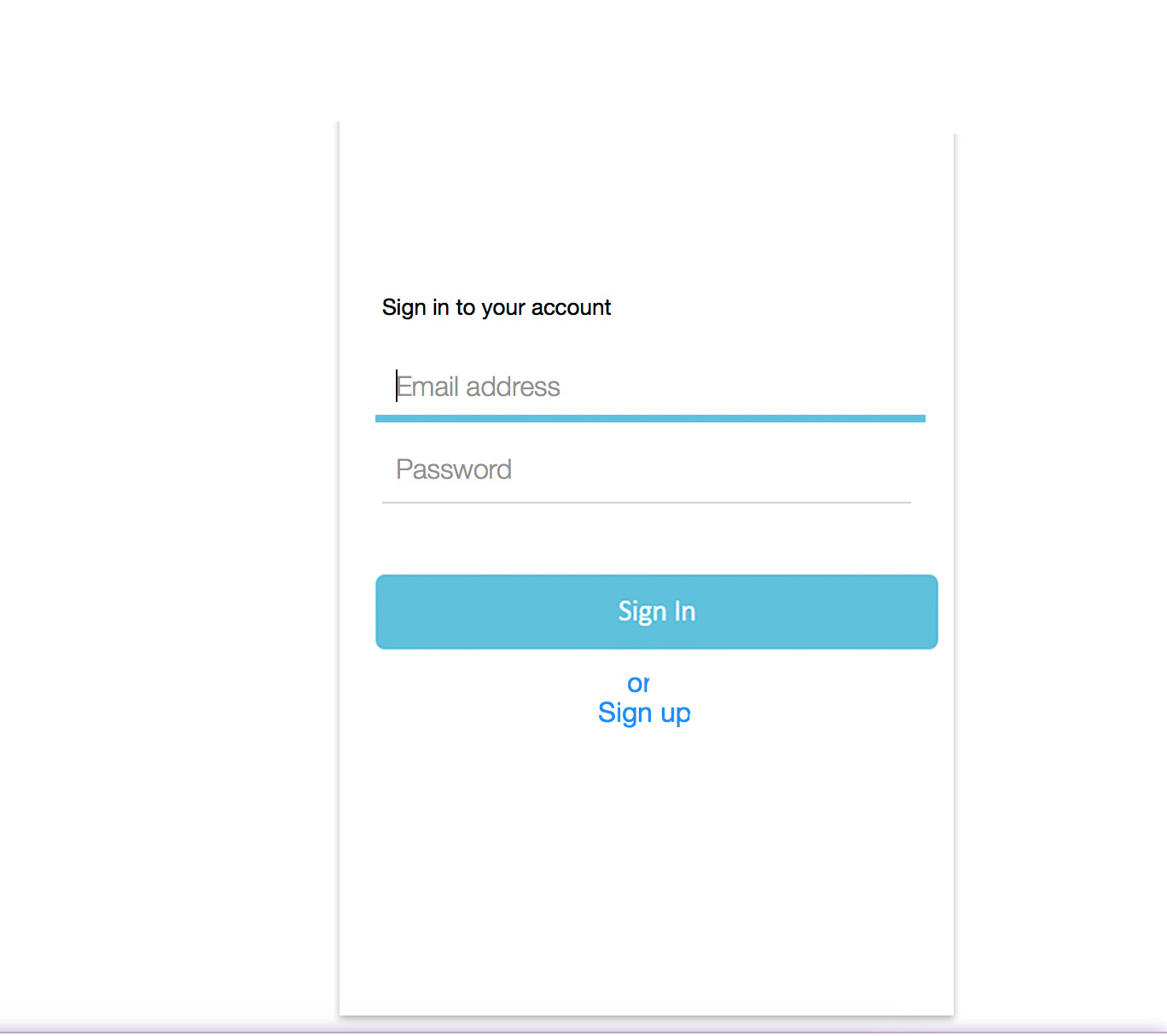


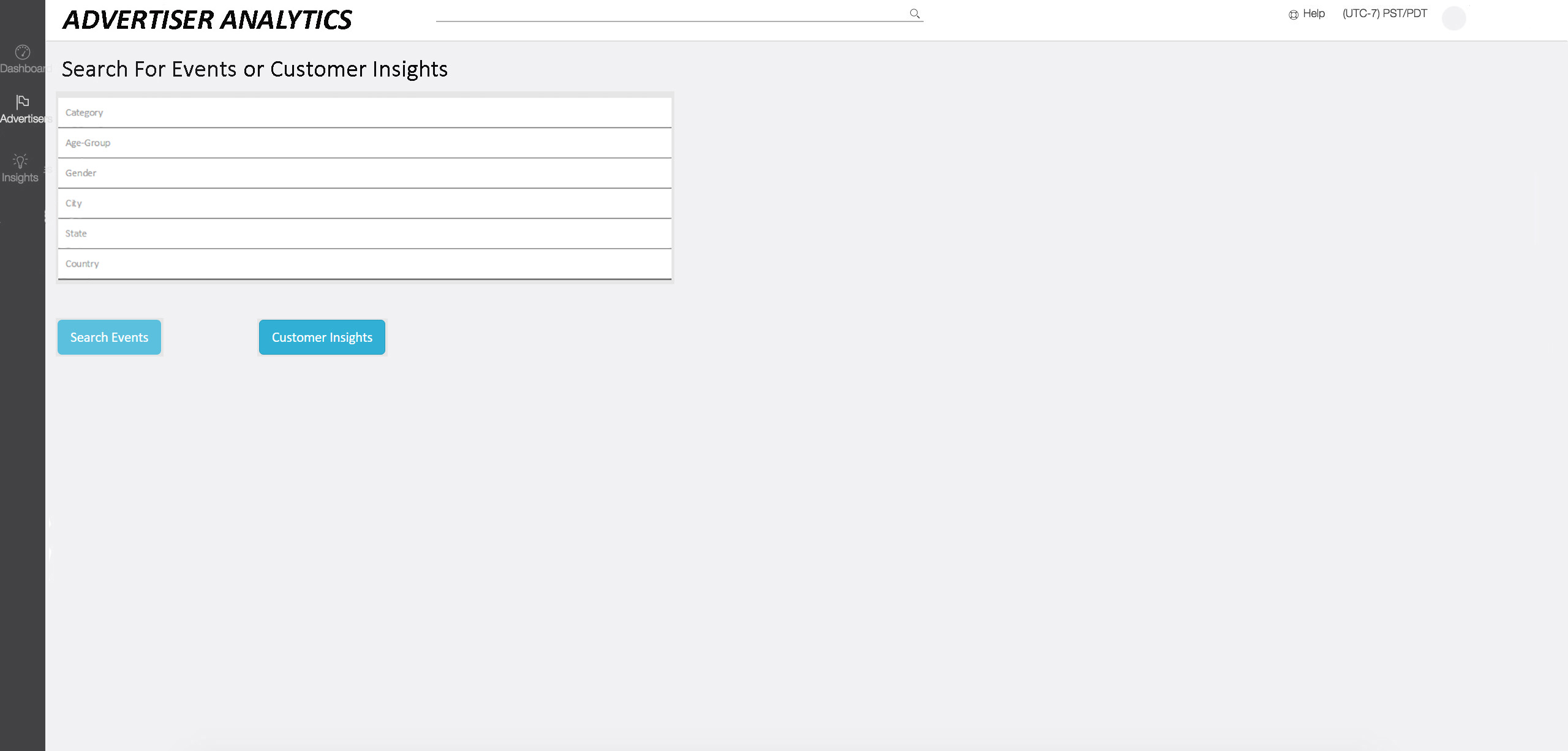












# QA, Performance, Deployment Plan

We have written Test plan for our application. We categorized our tests into Unit testing, Functional Testing, Integration testing, Usability Testing and Security. We used Test Driven development approach for developing our application.

We have used following testing frameworks in our application

* **Jasmine**

Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests.

* **Mocha**

Mocha is a feature-rich JavaScript test framework running on [Node.js](http://nodejs.org/) and the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. It is really simple to use and lets us keep our test files small and focused on one particular unit of code. Using the Behavior-driven Development (BDD) interface of Mocha also makes our tests very readable and obvious to anyone as to what is being tested.

* **Nock**

Nock is an HTTP mocking and expectations library for Node.js Nock can be used to test modules that perform HTTP requests in isolation. For instance, if a module makes HTTP requests to the Amazon API, you can test that module in isolation.

**Unit Testing:**

Unit testing is performed at the end of each and every module of our application to check whether each module is working properly or not. Developers will create and review Test cases.

Some of the unit test cases are:

* Verify if the data is getting pushed from the different API’s to the application.
* Check if the data from multiple API’s is getting synced at regular intervals to cloud.
* Check whether application is providing necessary recommendations to the end users (Advertisers, Advertising agencies, Individuals).

# Integration Testing

Integration Testing will be performed while integrating individual modules. All the unit tested modules will be integrated and tested. We are using bottom up approach for integrating modules and validating those modules.

Some of the test case that we will perform are:

* Integrate the UI with backend and test it.
* Validate if the application is able to receive the data from the multiple API’s and process it.
* Check for the User Interface provided by the application for collecting purchase activity, geographic and demographics of users.

**Functional Testing**

Functional testing is performed after developing the application.

|  |  |
| --- | --- |
| Test Case ID: 1 | Test Case Name: Advertiser Dashboard |
| Purpose | To provides audience insights which helps people (advertisers, ad agencies or individuals) make better use of their advertising budget and expand their market. |
| Pre-requisite | Processed data from public API’s. |
| Priority | High |
| Steps | 1. Collecting data from multiple API’s. 2. Process huge amount of data which is collected from multiple API’s by using Data Mining algorithms. 3. Based on the user request provides dashboard to end user. |
| Result | Provides Advertiser Dashboard to end users. |
| Tester Name | Santosh Sibyala, Srinivas Joga. |

|  |  |
| --- | --- |
| Test Case ID: 2 | Test Case Name: Uploading advertiser product details |
| Purpose | Based on advertiser product details we should request public API’s for user centric data. |
| Pre-requisite | Public API’s like Facebook and Twitter API’s. |
| Priority | High |
| Steps | 1. Developing user interface for uploading advertiser product details. 2. Requesting multiple API’s based on advertiser product details. |
| Result | User interface for collecting advertiser product details. |
| Tester Name | Venkat Kotu, Kiran Dwaram. |

**Usability Testing:**

Usability Testing is performed when our application is developed and validated. We will test User Interface and flow of pages of our application. In the usability testing we are planning to test our application capability to meet our intended purpose. To test the usability of our application we request our targeted users to test our application and based on their feedback we will make changes in our application.

**Performance Testing:**

At the application level and user level we will perform performance testing. We will test performance of data synchronization (data from different API’s to our database and data from our database to our application). We will also validate if the data synchronization is happening regularly and efficiently.

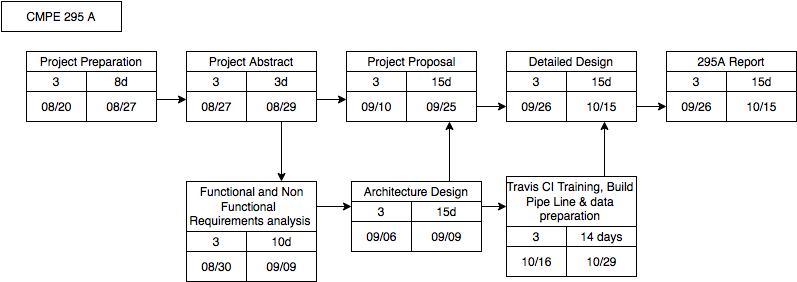
**Security:**

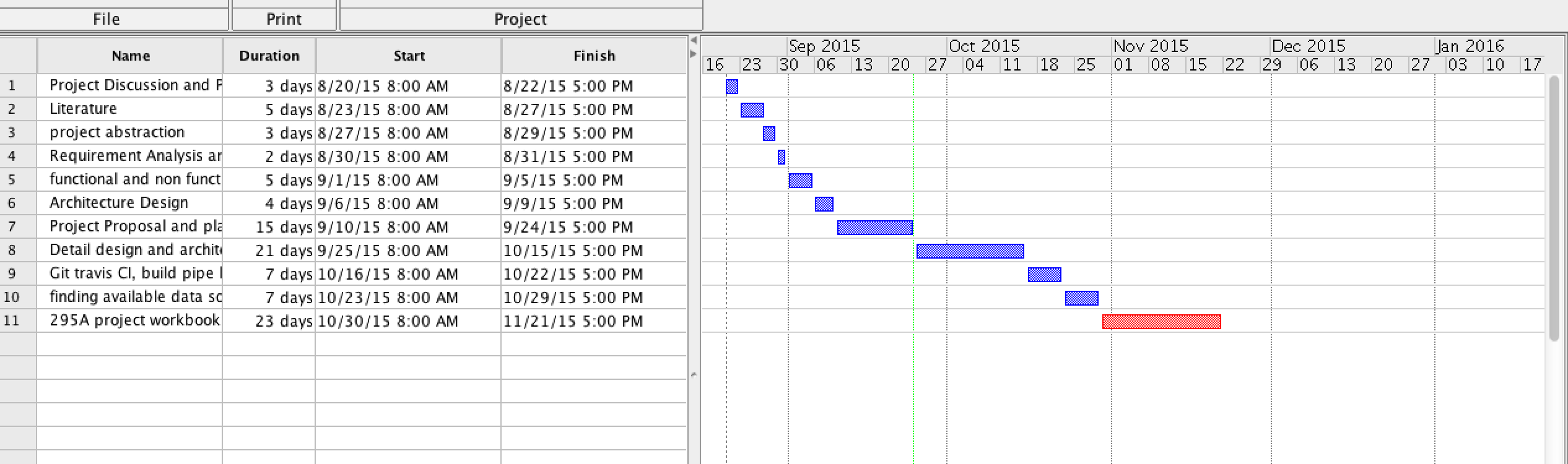
We want to make sure the data is secure. We are going to handle huge amount of sensitive data so, we will make sure that security is maintained well and we will encrypt the user data. **Implementation Plan and Progress**

|  |  |  |
| --- | --- | --- |
| **Implementation Plan** | **Resources** | **Status** |
| Environment setup for development | NodeJS, Eclipse IDE, GitHub, Mysql, Hadoop | Completed |
| Front End development tools setup | AngularJS, High charts, D3JS, Bootstrap, CSS3, HTML5 | Completed |
| Cloud infrastructure setup | AWS, Heroku | Completed |
| **Build Pipeline for continuous Integration** | **Travis CI, GitHub** | **Completed** |
| **Developing User Interface prototype with sample data.** | **Mockups** | **Completed** |
| **Designing the database schema** | **ERD plus** | **Completed** |
| **UML** | **ERD plus, genMyModel** | **Completed** |
| **Collecting the data required for project** | **Eventbrite api, ESPN api, Twitter, facbook api’s etc** | **In progress** |
| Identifying the right data mining algorithm to process data. |  |  |
| Processing data using data mining algorithm. |  |  |
| Storing the processed data in Relational Database. |  |  |
| Implementing requirements set. |  |  |

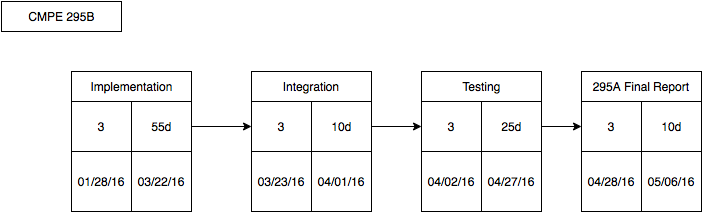
|  |  |  |
| --- | --- | --- |
| Unit Testing of functionalities. |  |  |
| Integrating both frontend and backend development parts. |  |  |
| Integration Testing |  |  |
| Deploying to Cloud. |  |  |

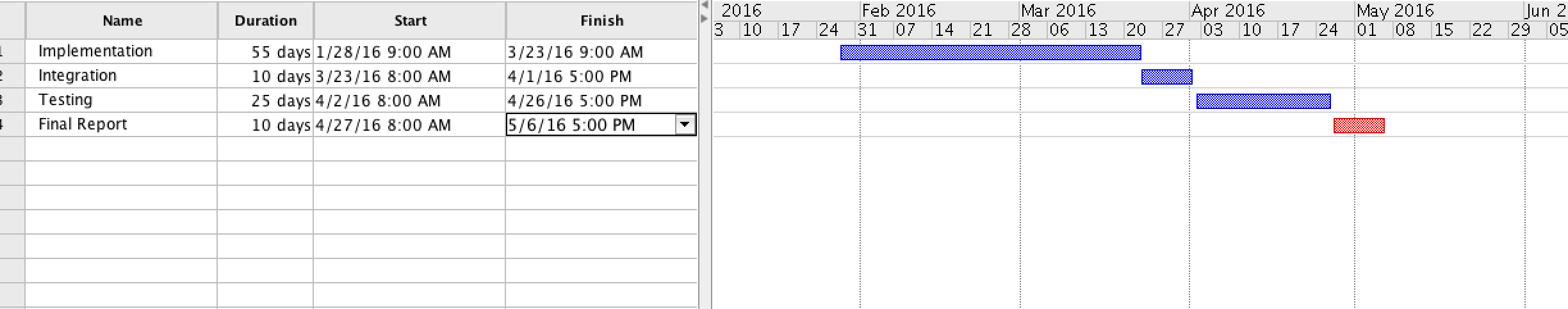
**Project Schedule**

****

****

**Table 5. Project schedule from Aug, 2015 to Dec, 2015**

****

****

**Table 6. Project schedule from Jan, 2015 to May, 2015**

**Table 7. Tasks and Resource Allocation**

CMPE 295A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Task Name | Duration | Start | Finish | Initials |
| 1 | Initial  Preparation | 8 days | 08/20/2015 | 08/27/2015 |  |
| 1.1 | Project discussion and plan | 3 days | 08/20/2015 | 08/22/2015 | VK,KD,JC,SS |
| 1.2 | Literature Preparation | 5 days | 08/23/2015 | 08/27/2015 | JC,SS |
| 2 | Project Abstract | 3 days | 08/27/2015 | 08/29/2015 | VK,KD,JC,SS |
| 3 | Requirements Analysis & Gathering | 2 days | 08/30/2015 | 08/31/2015 | VK,KD,JC,SS |
| 4 | Requirements - functional & Non functional | 5 days | 09/01/2015 | 09/05/2015 | KD,VK |
| 5 | Architecture Design | 3 days | 09/06/2015 | 09/09/2015 | VK,KD,JC,SS |
| 6 | Project proposal and plan preparation | 15 days | 09/10/2015 | 09/25/2015 |  |
| 6.1 | Introduction and Initial work | 5 days | 09/10/2015 | 09/15/2015 | JC,SS |
| 6.2 | Goals, Solutions , objectives & Schedule analysis | 10 days | 09/16/2015 | 09/25/2015 | VK,KD,JC,SS |
| 7. | Design and Architecture | 20 days | 09/26/2015 | 10/15/2015 |  |
| 7.1 | Technical arch design | 4 days | 09/26/2015 | 09/27/2015 | JC,SS,VK,KD |
| 7.2 | Database design | 4 days | 09/28/2015 | 10/01/2015 | VK,KD |
| 7.3 | Web Client UI design | 5 days | 10/02/2015 | 10/07/2015 | SS,JC |
| 7.4 | Backend Design | 7 days | 10/08/2015 | 10/15/2015 | VK,KD |
| 8 | Git, Travis CI build Pipeline training | 7 days | 10/16/2015 | 10/22/2015 | VK,KD,JC,SS |
| 9 | Finding available data sources | 7 days | 10/23/2015 | 10/29/2015 | VK,KD,SS,JC |
| 10 | 295A Project Report | 23 days | 10/30/2015 | 11/20/2015 |  |
| 10.1 | Project workbook draft | 10 days | 10/30/2015 | 11/08/2015 | VK,KD,SS,JC |
| 10.2 | Final Project Report | 12 days | 11/09/2015 | 11/20/2015 | VK,KD,SS,JC |

CMPE 295B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Task Name | Duration | Start | Finish | Initials |
| 1 | Implementation | 55 days | 01/28/2016 | 03/22/2016 |  |
| 1.1 | Database and data processing | 15 days | 01/28/2016 | 11/02/2016 | VK,KD |
| 1.2 | Backend Implementation | 45 days | 11/03/2016 | 03/22/2016 | VK,KD,SS,JC |
| 1.3 | UI implementation | 45 days | 11/03/2016 | 03/22/2016 | KD,JC,SS,VK |
| 2 | Integration | 10 days | 03/23/2016 | 04/01/2016 |  |
| 2.1 | DB & service integration | 5 days | 03/23/2016 | 03/27/2016 | JC,SS |
| 2.2 | Client Integration | 5 days | 03/28/2016 | 04/01/2016 | KD,VK |
| 3 | Testing | 25 days | 04/02/2016 | 04/27/2016 |  |
| 3.1 | Functional testing | 5 days | 04/02/2016 | 04/06/2016 | VK,KD,SS,JC |
| 3.2 | GUI testing | 3 days | 04/07/2016 | 04/09/2016 | VK,KD,SS,JC |
| 3.3 | Backend unit & system Testing | 7 days | 04/10/2016 | 04/16/2016 | VK,KD,JC,SS |
| 3.4 | Acceptance, Integration and End to End testing | 10 days | 04/17/2016 | 04/27/2016 | VK,KD,JC,SS |
| 4 | Final Report | 10d | 04/28/2016 | 05/06/2016 | VK,KD,JC,SS |