**System Design**

**Document**

*Piece of Eden Rentals*

**Client**

John Winder

**Team 4**

Stephen Masterson

Rachel Cohen

Nicholas Keckeisen

Eric Forte

Matthew Walker

3/5/2016

*Piece of Eden Rentals*

System Design Document

**Table of Contents**

1. Introduction

1.1 Purpose of This Document

1.2 References

1. System Architecture

2.1 Architectural Design

2.2 Decomposition Description

1. Persistent Data Design

3.1 Database Descriptions

1. Requirements Matrix
2. Appendix A – Agreement Between Customer and Contractor
3. Appendix B – Team Review Sign-off
4. Appendix C – Document Contributions

**1. Introduction**

***1.1 Purpose of This Document***

The purpose of this document is to describe the design of the Piece of Eden Rentals

application. Key topics covered in this document include the high level system

architecture, lower level class designs, and the persistent data design Piece of Eden Rentals.

***1.2 References***

Throughout this document references will be made to: (none at this time)

**2. System Architecture**

***2.1 Architectural Design***

The Piece of Eden Rentals application will be built using the Amazon Web Services framework (AWS). The core components will be build using the Django framework and a MongoDB database within AWS.

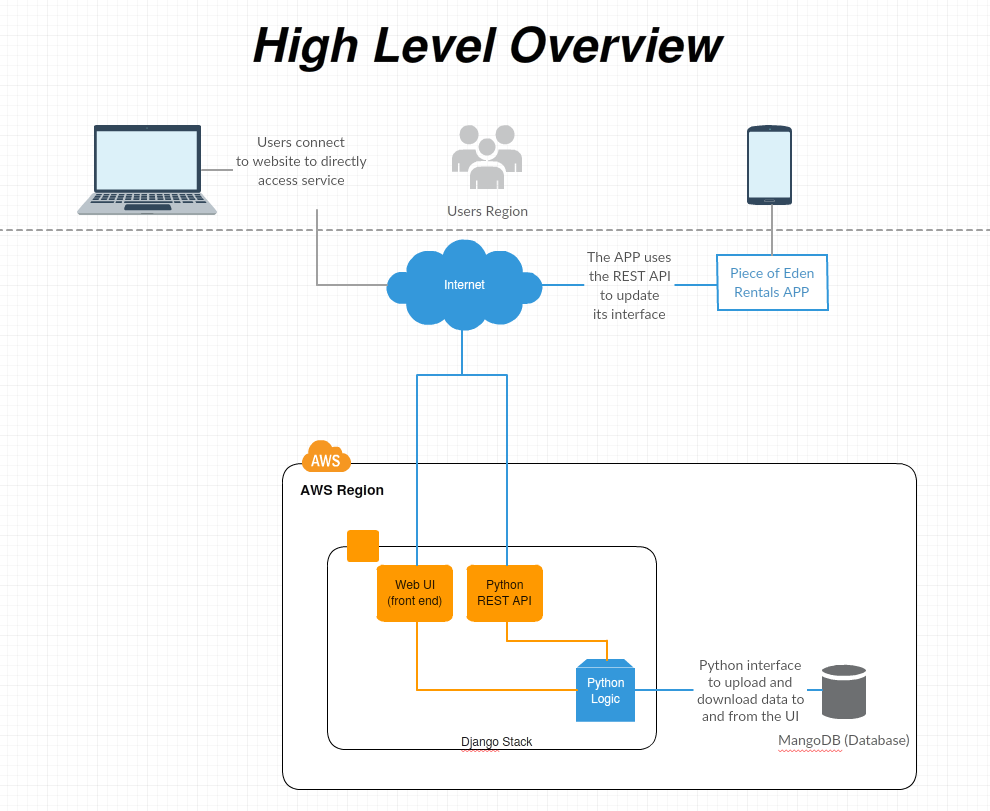


Figure 1.1

As shown in figure 1.1, users can either connect to the Piece of Eden Rental service directly through the web interface or on a mobile device through the APP. The APP uses a Python REST API integrated into the Django framework to communicate with the Database to obtain new information to display in its UI.

After this is still the template

--------------------------------------------------------------------------------------------------------------------

The basic components of the Django framework include Models, Views, and Templates, which follows a Model View Controller architecture (MVC). The model layer consists of classes, each of which refer to a specific table in a database. All data used in our application will be represented by a model, with the actual data existing in a database. Model classes describe either music

data or user data, and will be described in more details later in this document.

Every model class in our system has a corresponding table in our database. Our

persistent data design will be described later in the document.

Templates are the specific pages served to the user. Templates are made

up of html and other static files. The templates used in this application will be

described more fully in the User Interface Design Document.

The views are a layer of abstraction between the model and template layers. A

view, or the controller in MVC, is a function which takes input and returns output

to the user through the template layer. Normally the view would interact with

any number of model classes, then return either return some subset of data,

or perform operations on data and return the results. Views in this application

include search, view details, and browse, among others which will be described

more fully later in this document.

***2.2 Decomposition Description***

After this is still the template

--------------------------------------------------------------------------------------------------------------------

Each model in MusicBug is a Python class. The models related to music were

generated automatically by a Django tool, which based the models off of our

pre-existing legacy database (which is described later in this document). Each

model class defines the data fields as they appear in the database. All fields

are publicly accessible. The main model classes used in MusicBug are: Artist,

ArtistName, Release, ReleaseName, Track, TrackName, User, Review, and

Rating. The contents of these classes are simply the fields of the corresponding

database tables. The classes which hold music data are direct copies of the

tables from the MusicBrainz database. For example, The Artist class includes:

Name, Gender, BeginDate, Country, and other relevant artist data. The classes

which hold user data correspond to a separate user database which is populated

as the application is used. When a user registers with MusicBug, an entry in the

user table is populated, the fields for the User class include: Name, Birthdate,

RegisterDate, LastLoginDate, gender, location, and fid (from Facebook). The

Review, Rating, and Favorite classes simply contain the User, the date reviewed,

rated or favorited, the subject being rated, reviewed, or favorited, and the rating

or review itself.

Each view is a python function which takes an HTTP request and returns an

HTTP response. The search function returns the result of a query of all artist,

albums, and songs, matching a search term given by the user. The view details

function returns the details of an artist, track, or album. The login view registers

the user in the MusicBug system, and stores the user data in the database.

The Rate view stores the user’s rating of a track, album, or artist. The review

view stores the user’s review of a track, album or artist. The favorite view saves

an artist, track, or album in the user’s favorites. The comment view places a

comment generated by the user in the details page for an artist, track, album

or review. The view profile view shows the information we have pulled from

Facebook of the user, along with the user’s ratings, reviews, and favorites. The

register, favorite, and comment views each use the Facebook api to interact

with their Facebook account, the other views only interact with the models. Each

view directly implements a use case of MusicBug as described in the System

Requirements Document.

**3. Persistent Data Design**

***3.1 Database Descriptions***

An entity-relationship of our database schema for our music database can be

found on MusicBrainz website. The data we are using for MusicBug is provided

by MusicBrainz, which provides a free database of music meta-data. MusicBug

uses a SQLite version 3 database, which is a direct copy of the MusicBrainz

database.

The first step of importing the data from MusicBrainz was to create a new

database that emulated the MusicBrainz database structure. This was completed

by tweaking a SQL script which can be found in Lukáš Lalinský github project

(Lalinský 2012). Once the table structure was created, a script was written

to produce an import statement for every table in the data dump provided by

MusicBrainz. After running Lalinský’s creation script, and the generated import

script, the MusicBug database was created. The model abstraction classes were

then generated by a Django command. Built-in Django functions allow access to

the database using python instead of raw SQL.

All user database is kept in a separate database named Users. This database

was created automatically by the Django framework. After creating the user

related classes in the models portion of Django, a built in tool created the

database to our specification. The schema for the User database is modeled

below:

**4. Requirements Matrix**

Please refer to the System Requirements Specification for details regarding the

corresponding use cases.

**5. Appendix A – Agreement Between Customer and**

**Contractor**

The customer agrees to a *Music Social Network* system with searching, browsing and

detailed meta-data capabilities. See System Requirements Specification for more

information. Additional features will be provided in further development spirals.

When and if future changes to this document occur a drafted new document will be

created. Both a hard and electric copy of both versions will be presented to the client for

review. Upon approval, the draft will be finalized and signed off by both parties.

**Client**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

**Team**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

**6. Appendix B – Team Review Sign-off**

This document has been collaboratively written by all members the team.

Additionally, all team members have reviewed this document and agree on both

the content and the format. Any disagreements or concerns are addressed in

team comments below.

**Team**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Print

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature

Comments

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_

**7. Appendix C – Document Contributions**

Sean Ehrig took the lead on the writing of this document accounting for the

completion of 100%. Michael Cohen contributed to editing of the document.