eDocent

Group 5 Final Presentation

Project Goals

Create an Android application allowing museum patrons to learn more about works of art.

Create a Web application where museum administrators can provide information for the Android app.

Grow as software engineers and learn how to build a quality product as a group.

Museum & Patrons

Museum Administrators

Interact with our Django web interface, add works of art and museum information

Patrons

Interact with our Android application, find nearby museums, listen to audio clips about the art and scan QR codes to interact with artwork

Requirement Analysis

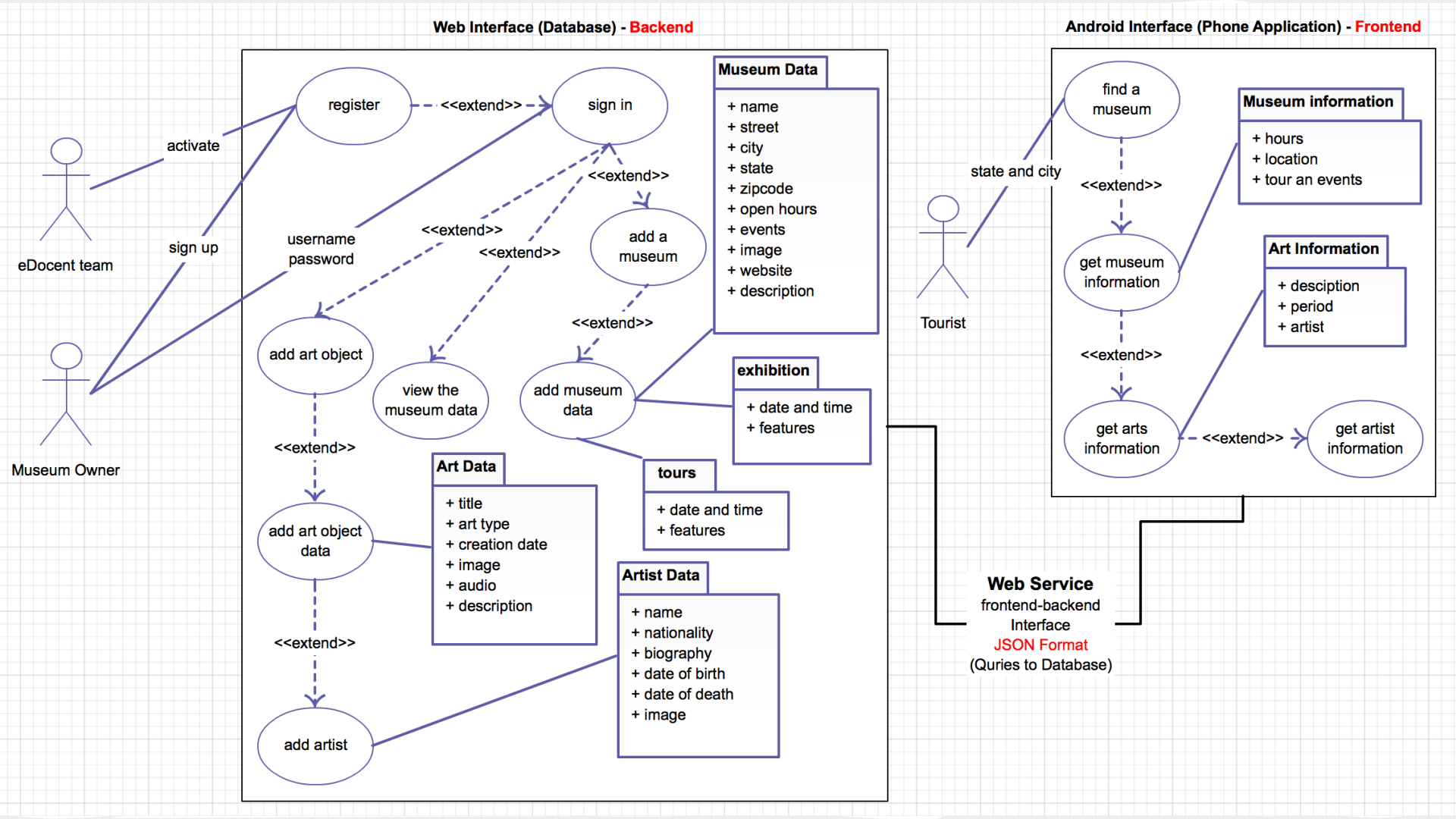
Functional Requirements

- Secure Museum Login
- Upload/edit/delete content
- Access content through mobile app
- Splash page to display museum info
- Ability to look through museum collection
- Lookup item through QR code

Use Case Diagram

- Frontend (Android Application)
  - One human actor - tourist
- Backend (Web Interface)
  - Museum’s owner actor
  - eDocent team actor

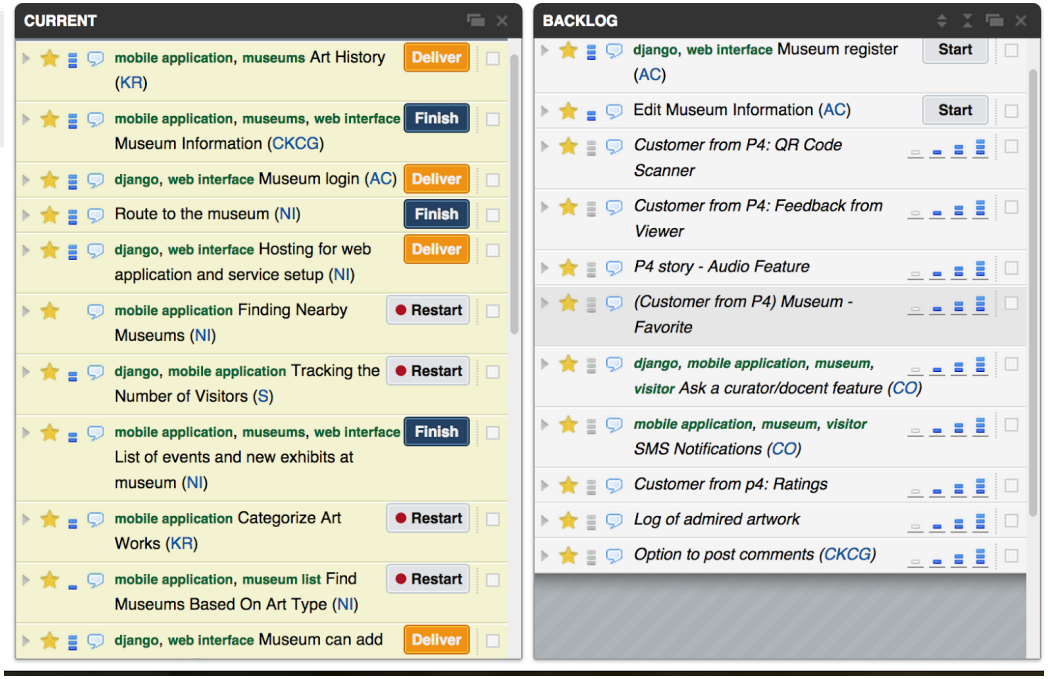
- (The Diagram Next Page)


Web Interface (Database) - Backend
Android Interface (Phone Application) - Frontend
eDocent team
activate
sign up
Museum Owner
register
sign in
<<extend>>
username
password
<<extend>>
<<extend>>
<<extend>>
add a
museum
<<extend>>
add art object
view the
museum data
add museum
data
<<extend>>
add art object
data
<<extend>>
add artist
Museum Data
+ name
+ street
+ city
+ state
+ zipcode
+ open hours
+ events
+ image
+ website
+ description
exhibition
+ date and time
+ features
tours
+ date and time
+ features
Art Data
+ title
+ art type
+ creation date
+ image
+ audio
+ description
Artist Data
+ name
+ nationality
+ biography
+ date of birth
+ date of death
+ image
Tourist
state and city
find a
museum
<<extend>>
get museum
information
<<extend>>
get arts
information
<<extend>>
get artist
information
Museum information
+ hours
+ location
+ tour an events
Art Information
+ desciption
+ period
+ artist
Web Service
frontend-backend
Interface
JSON Format
(Quries to Database)

User Stories

Pivotal Tracker

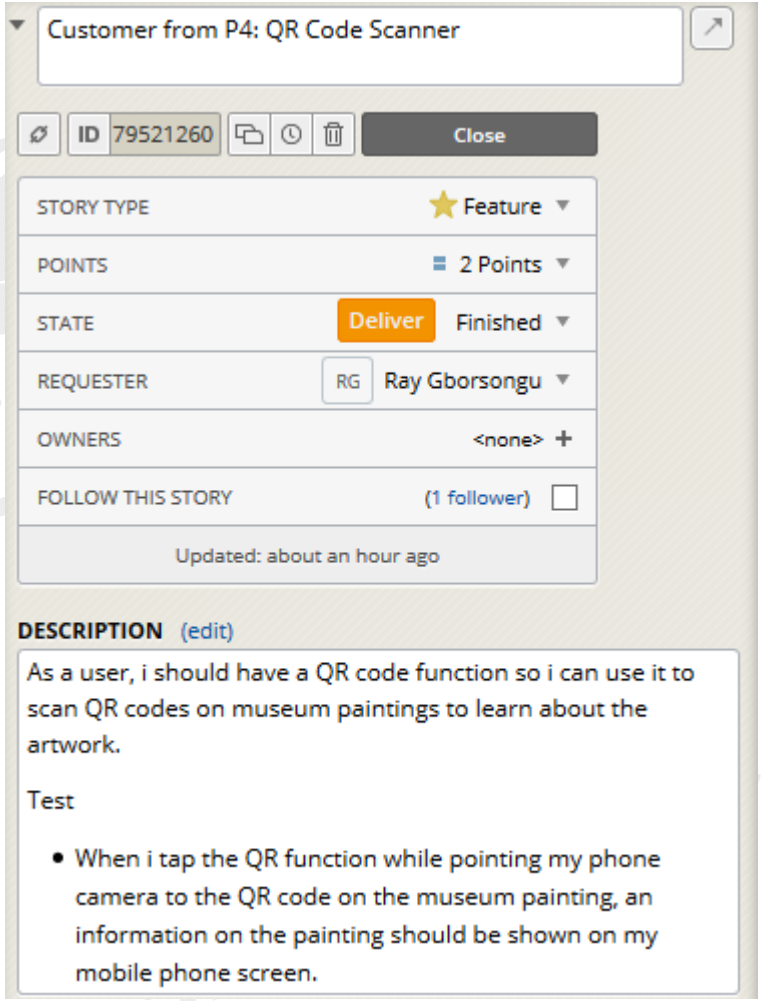
- Total number of user stories: 30
- Completed: 15
- Planned: 18



User Stories

Sample

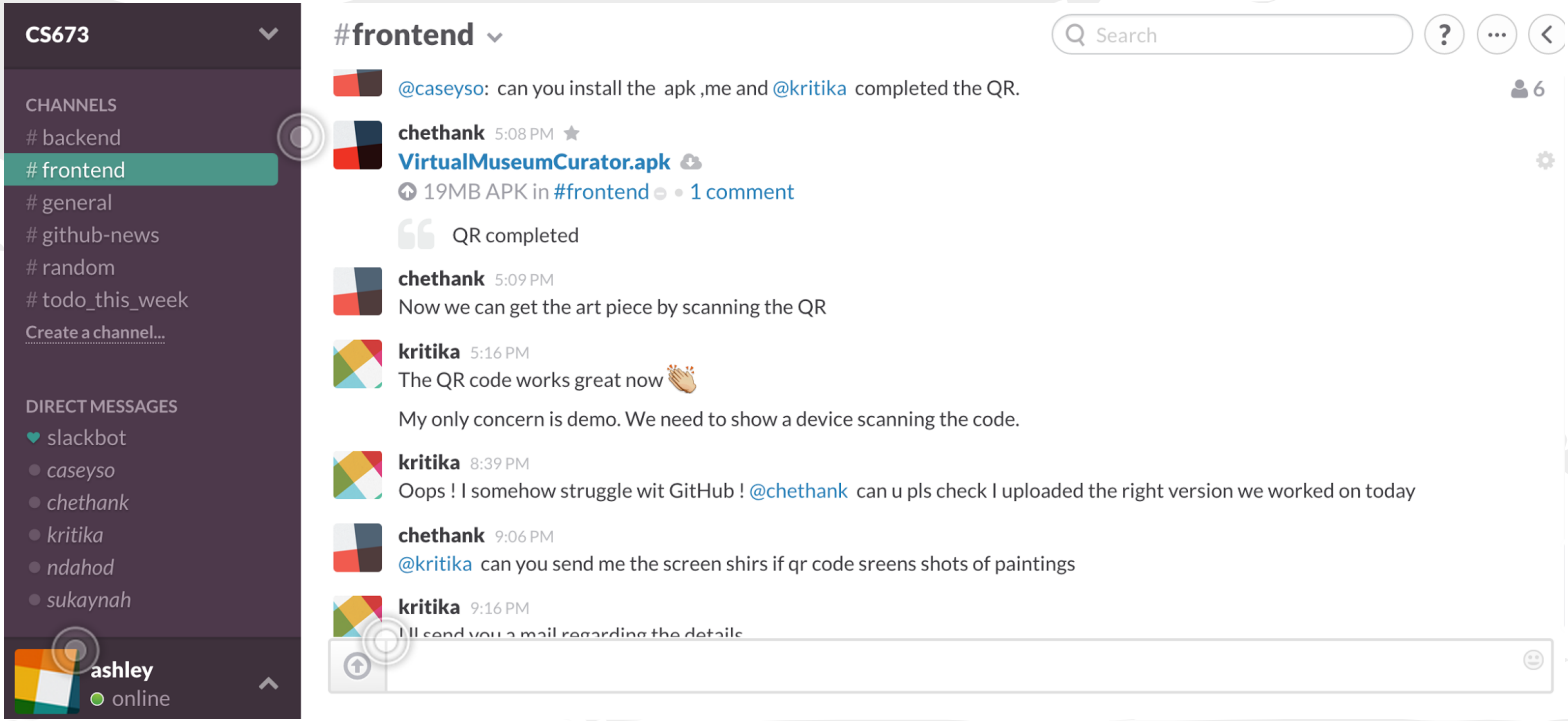
- Complete: QR Code
- Planned but not completed: Search by art work type
- Example of possible extension: Feedback/Comment for an art work



Requirement Analysis

Tracking requirements and changes

Slack - progress shared



Requirement Analysis

Tracking requirements and changes

GitHub - commit messages

3 days ago
Kritika688 opened issue tab262/cs673#15
Art info - Share on Social Media

3 days ago
Kritika688 pushed to master at tab262/cs673
`d88a130` Merge branch 'master' of https://github.com/tab262/cs673
`b1c1cc2` QR code final working
2 more commits »

ndahod created branch `Audio` at tab262/cs673 4 days ago

5 days ago
Kritika688 pushed to ArtWorkPage at tab262/cs673
`8d14484` adding image display files

6 days ago
tab262 opened issue tab262/cs673#14
Information section has formatting issues

12 days ago
chethan28 pushed to android at tab262/cs673
`05e022e` Collection page done done done yay yaya yay

13 days ago
chethan28 pushed to android at tab262/cs673
`81831f7` Splash page is dynamic

13 days ago
tab262 pushed to master at tab262/museum
`58d5cd6` model update

13 days ago
tab262 pushed to master at tab262/museum
`676c278` locations api changed

tab262 created repository tab262/lab05 14 days ago

15 days ago
tab262 pushed to master at tab262/museum
`f609b09` settings

Changes

Requirement changes

- Added features: Automatic latitude/longitude calculation from address in Django admin, social media sharing on Android app

How we tracked changes

- Introduced in weekly meeting or Slack conversation
- Status updated via GitHub commits and Slack discussion

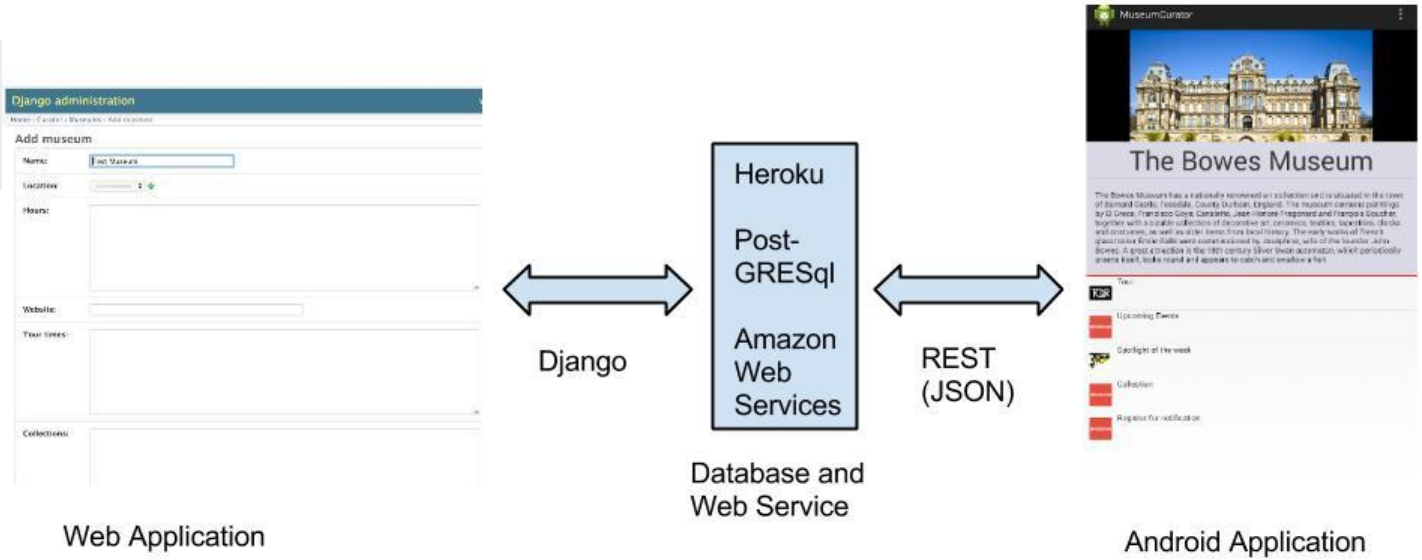
Non-functional requirements

- Intuitive museum admin form fields
- Museum admin filtering for user-friendly interface
- Intuitive user-interface of Android app
- Visually pleasing user-interface of Android app
- Security

Design

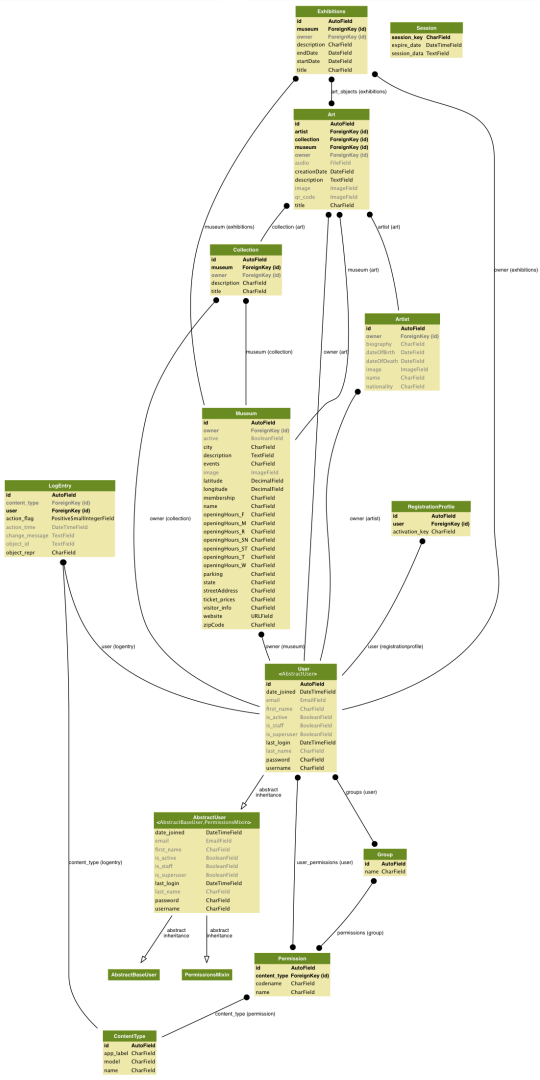
Design

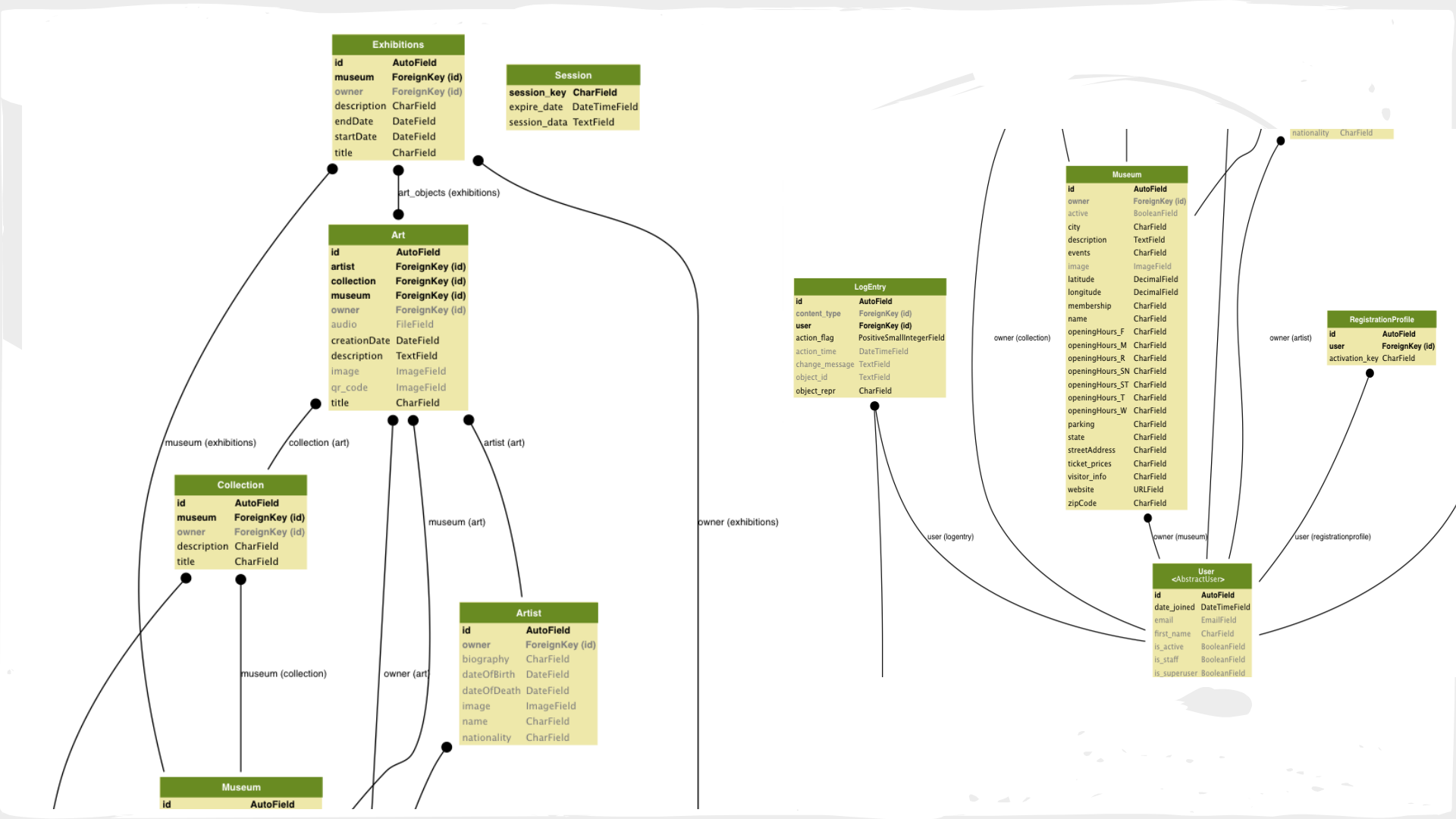
Software Architecture



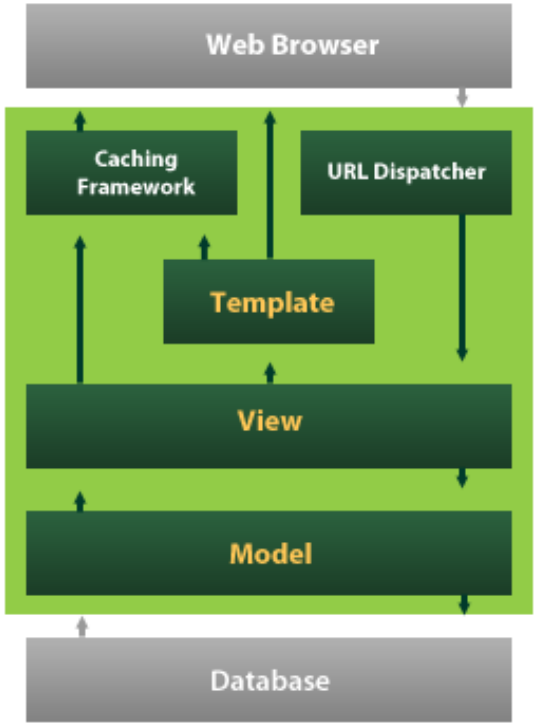
Design

- Database Modeling (ER-Diagram)
  - Museum
  - Art
  - Artist
  - Exhibitions
  - Collections



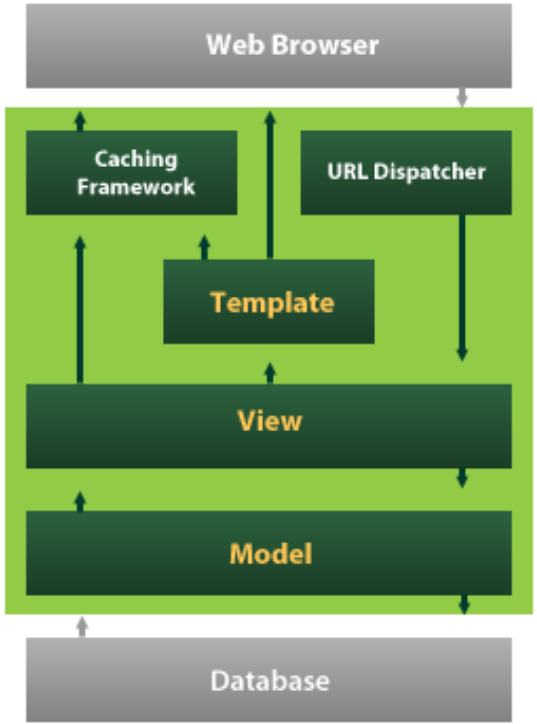

Exhibitions
id AutoField
museum ForeignKey (id)
owner ForeignKey (id)
description CharField
endDate DateField
startDate DateField
title CharField
Session
session_key CharField
expire_date DateTimeField
session_data TextField
art_objects (exhibitions)
Art
id AutoField
artist ForeignKey (id)
collection ForeignKey (id)
museum ForeignKey (id)
owner ForeignKey (id)
audio FileField
creationDate DateField
description TextField
image ImageField
qr_code ImageField
title CharField
museum (exhibitions)
collection (art)
artist (art)
museum (art)
owner (exhibitions)
Collection
id AutoField
museum ForeignKey (id)
owner ForeignKey (id)
description CharField
title CharField
museum (collection)
owner (art)
Artist
id AutoField
owner ForeignKey (id)
biography CharField
dateOfBirth DateField
dateOfDeath DateField
image ImageField
name CharField
nationality CharField
Museum
id AutoField
LogEntry
id AutoField
content_type ForeignKey (id)
user ForeignKey (id)
action_flag PositiveSmallIntegerField
action_time DateTimeField
change_message TextField
object_id TextField
object_repr CharField
user (logentry)
owner (collection)
Museum
id AutoField
owner ForeignKey (id)
active BooleanField
city CharField
description TextField
events CharField
image ImageField
latitude DecimalField
longitude DecimalField
membership CharField
name CharField
openingHours_F CharField
openingHours_M CharField
openingHours_R CharField
openingHours_SN CharField
openingHours_ST CharField
openingHours_T CharField
openingHours_W CharField
parking CharField
state CharField
streetAddress CharField
ticket_prices CharField
visitor_info CharField
website URLField
zipCode CharField
owner (museum)
nationality CharField
owner (artist)
RegistrationProfile
id AutoField
user ForeignKey (id)
activation_key CharField
user (registrationprofile)
User
<AbstractUser>
id AutoField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField

Design - Django MVC



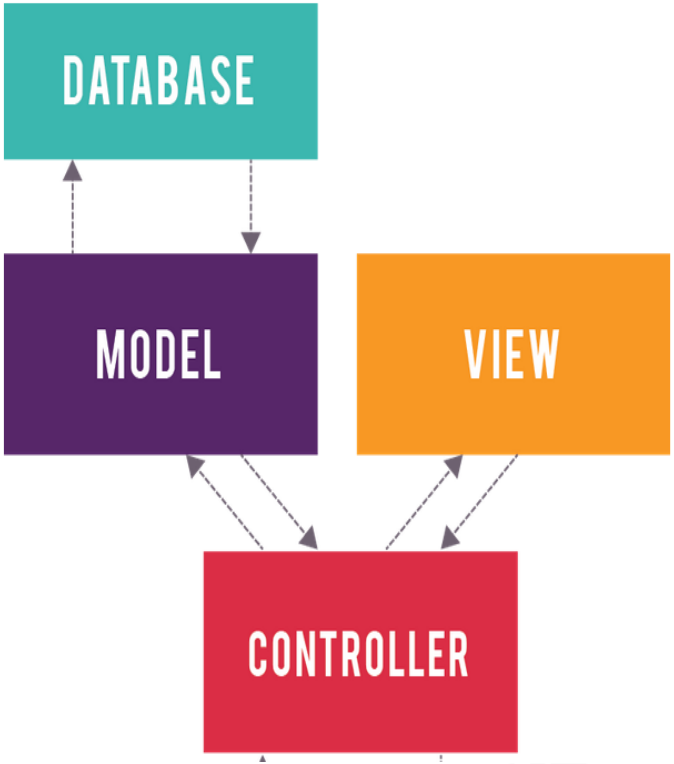
- **Model** - data-access, handled by Django's database layer
- **View** - selects which data to display and how to display it (handled by views & templates
- **Controller** - delegates to a view depending on user input, handled by the Django framework itself (URLconf)

Design - Django MTV



- **Model** - data access layer, contains everything about the data
- **Template** - presentation layer, how things are displayed
- **View** - business logic layer, bridge between models and templates

Design - Android MVC



View: Displays the application data to the user and gets data from the user.

Model: Responsible for maintaining the data. It communicates to the database through a RESTful API.

Controller: Intermediary between the View and the Model.

Implementation

Implementation

Tools

Challenges

- Finding enough time to work on the project and juggle busy schedules.
- Experience levels were different.
- Working in two groups and understanding the progress in each group.
- Adapting to different work styles

Implementation

Web Application

Features:

- Museum can login with username and password
- Museum can specify museum name, address, tour times, hours open, and more (address automatically converted to lat-lon for Android app's map feature)
- Museum can add works of Art, Artists, and Exhibitions
- Museums can only see items associated with their museum

Implementation

Android Application

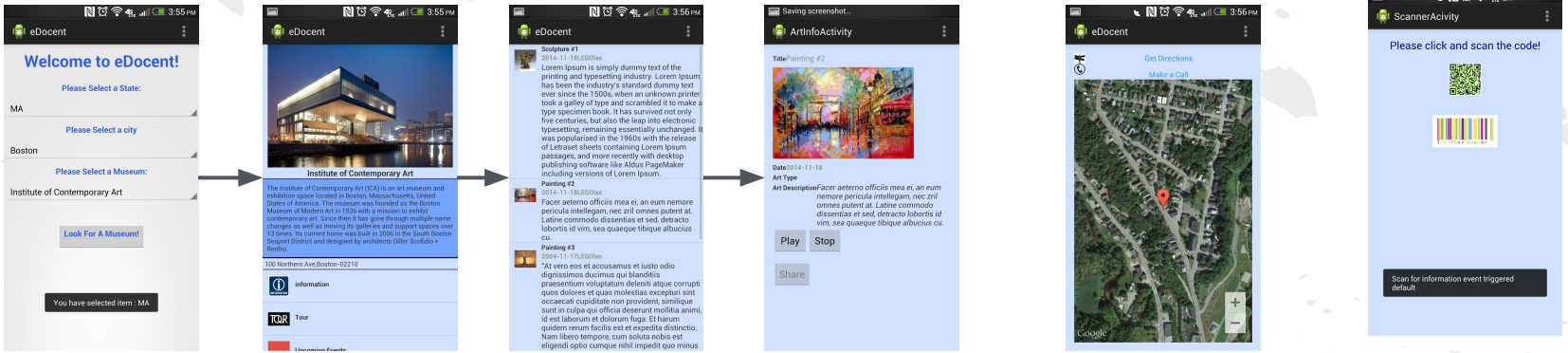
Features:

- User can choose a museum after they input their state and city.
- Directed to the museum home page which displays the picture of the museum, the hours, and the address. Users can also scroll through the options of taking a virtual tour, viewing upcoming events, viewing the ‘spotlight of the week,’ collections currently in the museum, events, maps (Google Maps), and the option to receive notifications from the museum.
- If the user selects ‘Maps,’ they will be connected to Google Maps and can get directions to the museum.
- If you click QR, then you can scan the code and be led to the art piece.
- If the user selects ‘Collections,’ they will see a list of all the art pieces in the museum’s collection.

Android Application

Features continued:

- They can select on an art piece and be led to a page, which displays a larger picture of the art piece, can play an audio clip about the piece, has information about the art piece, and can be shared on social media.



Implementation

Refactoring

Back end:

- Bad smell examples:
  - duplicated code
  - speculative generality

models.py

```
55
56  class Artist (models.Model):
57      name = models.CharField(max_length=200)
58      nationality = models.CharField(max_length=200)
59      biography = models.CharField(max_length=200)
60      dateOfBirth = models.DateField('DoB', null=True)
61      dateOfDeath = models.DateField('DoD', null="True", blank="True")
62      image = models.ImageField(upload_to="artist_images", blank=True)
63      owner = models.ForeignKey(User, null=True, blank=True)
64
65
66
67      def __unicode__(self):
68          return self.name
69
70  class Collection (models.Model):
71      title = models.CharField(max_length=200)
72      description = models.CharField(max_length=2000)
73      museum = models.ForeignKey(Museum)
74      owner = models.ForeignKey(User, null=True, blank=True)
75
76      def __unicode__(self):
77          return self.title
78
79
80  class Art (models.Model):
81      title = models.CharField(max_length=200)
82      artist = models.ForeignKey(Artist)
83      creationDate = models.DateField('creation date')
84      image = models.ImageField(upload_to="art_images", blank=True) #models.CharField(max_length=20000)
85      audio = models.FileField(upload_to="audio_files", blank=True)
86      description = models.TextField()
87      museum = models.ForeignKey(Museum)
88      owner = models.ForeignKey(User, null=True, blank=True)
89      #image2 = models.ImageField(upload_to="upload_image_to", blank=True)
90      qr_code = models.ImageField(null=True,upload_to="qr_codes", blank=True)
91      collection = models.ForeignKey(Collection, default=None)
92
93      def __unicode__(self):
94          return self.title
95
96      def qr_code_fn(self):
97          return '<img src="%s", height="150",width="150"/>' %  self.qr_code.url
98
99      qr_code_fn.allow_tags = True
100
101
102
103
104 class Exhibitions (models.Model):
105     title = models.CharField(max_length=200)
106     startDate = models.DateField('Exhibition Starts')
```

models.py 60,52 Python UTF-8 master +7, -10

Implementation

Refactoring

Front End:

- Bad smells:
  - Switch statements
  - Some duplicated code

```
  @Override
  public void onListItemClick(ListView l, View v, int position, long id) {
// do something with the data

    String item= (String) getListAdapter().getItem(position);

    switch(position){

    case 0:
        Intent intentInfo=new Intent(this.context,MuseumInformation.class);
        intentInfo.putStringArrayListExtra("hours", this.hours);
        intentInfo.putExtra("membership",this.membership);
        intentInfo.putExtra("parking",this.parking);
        intentInfo.putExtra("ticketprice",this.tickteprice);
        intentInfo.putExtra("website",this.website);
        intentInfo.putExtra("visitorsinfo", this.visitorsinfo);
        startActivity(intentInfo);
        break;


    case 4:
        Intent intentC=new Intent(this.context,CollectionPageActivity.class);
        intentC.putExtra("primarykey", this.primarykey);
        Log.d("Before CollectionPageActivity activity","********");
        startActivity(intentC);
        Toast.makeText(getActivity(), item+" Collection triggered", Toast.LENGTH_LONG).show();
        break;

    case 6:
        Intent intent =new Intent(this.context,MuseumLocationActivity.class);
        intent.putExtra("Address", this.Address);
        intent.putExtra("phone", this.phone);
        intent.putExtra("title", this.title);
        intent.putExtra("lat", this.lat);
        intent.putExtra("lng", this.lng);
        Log.d("Before start activity",this.Address+this.title+this.phone+this.lat +this.lng);
        startActivity(intent);
        Toast.makeText(getActivity(), item+" Map triggered", Toast.LENGTH_LONG).show();
        break;

    case 7:
        //Intent intentScan=new Intent("com.google.zxing.client.SCAN");
        Intent intentScan =new Intent(this.context,ScannerAcivity.class);
        startActivity(intentScan);



    default:
            Log.d("No activity","********");
                Toast.makeText(getActivity(), item+" event triggered default", Toast.LENGTH_LONG).show();

    }
```

Testing

Testing - Backend

- Backend
  - Manual
  - Unit testing
  - Regression testing
  - Test driven development

Testing - Frontend

- Manual testing
- Unit testing
- Automation framework
  - Robotium

Example of Unit Test Code for audio feature:

```
public void testSplashPage() {
        solo.assertCurrentActivity("testing splash page", EDocentSplashActivity.class);
        solo.pressSpinnerItem(0, 0);
        solo.pressSpinnerItem(1, 0);
        solo.pressSpinnerItem(2, 0);
        solo.clickOnButton("Look For A Museum!");
    }
```

Project Management

Project Management

- Backend Team Contributions:
  - Ashley:
    - Django admin developer (models, permissions, user experience)
    - Coordinator of group calendar, communication tools, planning
  - Sukaynah:
    - Designing diagrams (er-diagram, use case diagram)
    - Registration of new users & resetting password
  - Casey:
    - Django admin and REST API developer
    - Hosting configurations with Heroku and AWS
    - Testing

Project Management

- Frontend Team Contributions:
  - Chethan:
    - Splash Page
    - Museum page, Collections page
    - Art page -QR and image
  - Kritika:
    - Art page, QR, image, and audio
    - Collections page
    - Testing
  - Nisreen:
    - Splash page
    - Art page- audio
    - Testing

Project Management

Risk Management:

- Anticipated risks: miscommunication, time shortage, design errors, server crash, lack of testing, lack of technical understanding, loss of group member(s)
- Realized risks:
  - time shortage - very busy school schedules for all group members and important family and life events

Project Management

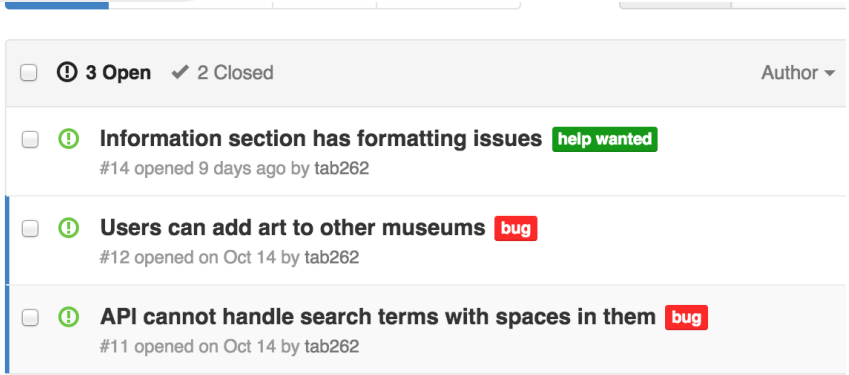
Quality Management:

- Ensure to accomplish what we promised our ‘customer.’
- Implementing defect detection techniques and focusing on fixing them.

Project Management

Quality Metrics:

- Fault Profile: Bugs and defects recorded on GitHub
- Breadth of testing: test all functionality.



Some recorded bugs on GitHub.

Project Management

Achievements:

- Delivering a project with all the features we promised our ‘customer.’
- We all learned new techniques and even learned to program in new platforms and environments
- Worked efficiently as two separate teams

Project Management

Challenges and Lessons learned:

- Remembering to log bugs and defects, especially small ones.
- Version control, different working environments and integration
- Database schema changes

Demo

Thank you!