IA8470
Project 2: Server-side API creation and Client-side Integration
Assigned: Monday 11/1/16
Due Dates:
(Materials) Tuesday Dec 13th (12/13/16) before class time (5:30pm)
Presentation: Tuesday Dec 13th (12/13/16)) during class time (final)

Overview

Project 2 continues the web development and design initiated in Project 1. You will create a Django REST API for servicing CRUD (create, read, update, delete) requests coming from your clientside ember app. Project 2 also offers you the first real opportunity to start realizing core security functionality. You will deploy your REST API on an Apache server and harden it against various forms of attack. Among other things you will harden your API by adding validators for each create or update operation and ensure that there are no side-effects from your methods. Additionally, you will update your design documents to reflect your server-side changes.

Team Grading Policy

Teams with more than one person on them will be required to submit confidential evaluation forms at the end of the project for themselves and their team members. Individual grades for those on teams will be calculated as follows:

Team grade X *participation factor = Individual grade*

Participation factor is a factor from .4 to 1 that is based on team evaluations, commit history on bitbucket, instructor communications, and ability to answer impromptu instructor questions about the project. Basically if you do your job and fully participate you will get a factor close to or equal to 1 (full points), if you don't you will get a factor proportional to your participation.

Total Points

1200 Points

Project Documentation (90 points)

You will be expected to turn in revised documentation based on instructor comments and any changes that resulted from application development activities in this project. The following rubric describes how your revisions will be graded in your submission.

Executive Summary and Risk List Revisions (30pts)	**Meets (13-15 pts)**	**Some Issues (8-12 pts)**	**Does not meet (0-7 pts)**
Integrates and Complies with Instructor Comments For Summary	All comments are addressed and smoothly integrated into the executive summary. Quality improvements are clear.	An effort to integrate instructor comments, but some issues still remain	Little to no effort made to resolve issues raised by instructor comments.
Integrates and Complies with Instructor Comments For risk list	All comments about identified risks are addressed and smoothly integrated into the risk table. Quality improvements are clear.	An effort to integrate instructor comments, but some issues still remain	Little to no effort made to resolve issues raised by instructor comments.
Use Case Revisions (30pts)	**Meets (13-15 pts)**	**Some Issues (8-12 pts)**	**Does not meet (0-7 pts)**
Integrates and Complies with Instructor Comments	All comments are addressed and smoothly integrated into the use case listings and use case diagram. Quality improvements are clear.	An effort to integrate instructor comments, but some issues still remain	Little to no effort made to resolve issues raised by instructor comments.
Additional Changes based on Project 2 activities	Use case diagram matches product functionality based on project 2 development efforts.	An effort made to adjust use case diagram, but some discrepancies between application and use case diagram remain	Little to no effort made to update use case diagram based on product 3.
Architecture Revisions (30pts)	**Meets (13-15 pts)**	**Some Issues (8-12 pts)**	**Does not meet (0-7 pts)**
Integrates and Complies with Instructor Comments	All comments are addressed and smoothly integrated into application and technical architecture diagrams. Clear effort to provide a quality final submission.	An effort to integrate instructor comments, but some issues still remain	Little to no effort made to resolve issues raised by instructor comments.
Additional Changes based on Project 2 activities	Architecture diagrams are adjusted to match product functionality based on project 2 development efforts.	An effort made to diagrams, but some discrepancies between application and use case diagram remain	Little to no effort made to update diagrams based on product 3.

Django API Functionality (540 points)

Implement an API to serve data to your client-side application. Your API should use Django REST Framework, Tastypie, django-restless, or **ANOTHER APPROVED FRAMEWORK**. I **strongly** suggest you use Django REST Framework as covered in class. Your API must demonstrate an understanding of data needs for your application as well as good use of security practices. The goal is to have an API that supplies all needed data and server-side functionality (e.g. authentication) to your ember app.

You are expected to:

(1) Create at django models to support your ember app (should replace all http-mocks)
   a. Data supports identified use cases with appropriate data fields
   b. Uses *View, add, update, delete* permissions as necessary
   c. Create JSON serializers for the models
   d. Create views and urls that support API calls for GET/POST/PUT/DELETE methods as necessary

(2) Support good authentication and permission policies
   a. Use session or token-based authentication for your REST API
   b. Apply global and/or local permission policies to REST API
   c. Has at least 4 django-auth user groups and assign them permissions.
      i. Superuser (logged in, has all privileges, this is given to you by django)
      ii. Application admin (logged in, has all privileges for app specific data)
      iii. Application user (logged in, has basic user privileges for the app)
      iv. Anonymous user (not logged in, has limited or no privileges for the app)

(3) Validate and sanitize all model data before saving
   a. Use django's clean and validators=[validate_function] methods as necessary
   b. Every model field escapes HTML as appropriate (if raw HTML is needed, reasoning is provided)
   c. Every model field sanitizes javascript (by removing it typically)
   d. Other validations used as needed (E.g. email validation, zip code, etc)

Models (200pts)	**Meets (41-50 pts)**	**Some Issues (20-40 pts)**	**Does not meet (0-19 pts)**
Appropriate fields	Model data fields fully support identified use cases and ember needs.	Model fields don't fully support use case and ember needs.	Model fields don't support identified use cases at all.
View permission defined	Model includes a view permission in its class Meta information.	Attempt made at view permission, but not correct.	No attempt at creating view permissions.
JSON Serializer	A serializer is defined either in the view or in a separate file to map the model fields into JSON Objects. Serializer produces valid JSON.	A serializer is defined either in the view or in a separate file to map the model fields into JSON Objects. Serializer is incorrect or doesn't always produce valid JSON.	No attempt made at serialization or very bad attempt made.
Views & URL wiring	Views and URLs support GET, POST, PUT, and DETETE on the model data as necessitated by your app.	Not all functionality is handled by views and urls for the model.	Many issues with views or url wiring prevents the API from functioning correctly.
Auth. and Perms. (120pts)	**Meets (16-20 pts)**	**Some Issues (9-15 pts)**	**Does not meet (0-8 pts)**
Session or token-based auth	Application uses session or token auth. and makes sessions/tokens available in	Application uses explicit session-based auth as defined in settings.py but sessions are	Application does not use explicit session-based auth.

	the admin interface.	not available in admin interface	
Global Permissions	Application uses DjangoModelPermissions by default.	Application uses another weaker permissions set such as IsAuthenticated.	Application uses AllowAny by default.
Super user group	Application includes a super user group.	N/A	Application does not include a super user group.
Application admin	Application includes an application admin group with appropriate permissions to access, modify, and delete any data in webapp but not in django-auth	Application includes an application admin group, but there are minor permission issues.	Does not contain an application admin group or permissions are completely wrong.
Application user	App. includes an application user group with appropriate permissions for app usage.	Application user group exists, but has minor permission issues.	No application user group or permissions are completely wrong.
Anonymous user	App. includes a user group for anonymous unauthenticated users. Minimal permissions are assigned as appropriate.	App includes a user group for anonymous users, but there are minor permission issues.	No anonymous user group or permissions are completely wrong.
Validation and Sanitization (120pts)	**Meets (32-40 pts)**	**Some Issues (15-31pts)**	**Does not meet (0-14 pts)**
Escapes HTML	All HTML is escaped or rationale is provided as to why raw HTML is needed. Escapes are applied to all html-capable fields (i.e. char, text, etc)	Most HTML is escaped, possibly some issues with escape method leads to possible vulnerabilities.	Raw HTML is not escaped and model fields are vulnerable to POST/PUT attacks.
Sanitized Javascript	All fields are sanitized against javascript entry. API calls cannot be used to inject javascript into the database.	Most javascript is sanitized, but there are small issues that may leave the app vulnerable.	Javascript is not sanitized, and can be demonstrably used for attack via POST/PUT.
Validator use	Validator and _clean method are used appropriately for custom validations (such as email, zipcode, or ssn fileds) to ensure fields are well formed.	Most validator use is well founded and applicable, but some obvious needs are missed.	Poor or no use of validators where needed
API maturity (100pts)	**Meets (80-100 pts)**	**Some Issues (50-79 pts)**	**Does not meet (0-49 pts)**
Maturity	Application is at an appropriate stage of development for the time frame.	The application is not as developed as it should be given the timeframe	Low effort to effectively prototype the app shows with a very low maturity app.

Ember Integration (150 pts)

Ember Integration	Meets (41-50 pts)	Some issues (20-40 pts)	Does not meet (0-19 pts)
Integrates with Django REST API	Ember client-side app integrates well with the Django REST API using Ember Data and/or custom jQuery Ajax calls. App is able to do all CRUD operations on data as necessitated by use cases.	Some issues with the integration limit or prevent certain CRUD operations.	Ember app does not integrate with Django REST API or there are many issues with the integration.
Implements a client-side session or token-based authentication mechanism	Allows different types of authenticated users to use suitable functionality in the app given use cases. Also stores session or ouath token appropriately and makes good use of session/local storage.	Some issues with the auth system prevent access for certain types of users or do not appropriately handle session variables.	Many issues with the auth system.
Displays errors and updates UI based on loaded Data	Ember app displays appropriate data in appropriate UI elements. E.g. updating forms to show errors returned from Django REST API.	Some issues displaying loaded data or errors are not all displayed correctly.	No or little attempt to display loaded data appropriately.

Apache Hardening (150 pts)

Ember Integration	Meets (41-50 pts)	Some issues (20-40 pts)	Does not meet (0-19 pts)
Apache Conf File	The apache conf file is configured consistent with best practices.	Some issues exist in the configuration that leave certain vulnerabilities open.	Conf file is poorly configured.
IPtables	Only approved connections are allowed through the iptables configuration.	Some holes exist, but attempts were made to configure appropriately.	Many holes exist allowing multiple forms of unwanted traffic.
Apache-django connection	Apache and Django are configured in a secure way – e.g. Django application errors are not displayed to users and wsgi is configured correctly.	Some apache-django issues are evident.	Many configuration details about Django are leakable to the web-facing user interfaces.

Github usage (50 pts)

It is very important that you use git and github THROUGHOUT project 2. Every time you make a significant change you should make a commit with an appropriately succinct but informative commit message. You should push commits to github after making them.

github	Meets (22-25 pts)	Some issues (12-21 pts)	Does not meet (0-11 pts)
Commits	Reasonable number of commits given time interval.	Far too many or far too few commits for the time interval.	No or almost no use of source code management.
Usernames and commit messages	Each commit includes a succinct informative commit message and is pushed to github using the username of the person who wrote the code (I don't want to see all commits from the same person on a team).	Most commits include succinct informative commit messages.	No commit messages, messages not informative, messages not succinct, or commits all coming from a single user account.

Project 2 Presentation: (100points)

Presentation	Meets (8-10 pts)	Minor Issues (6-7 pts)	Some issues (4-5pts)	Does not meet (0-3 pts)

Enthusiasm and clarity	Presenter exhibits enthusiasm about the work and speaks clearly and confidently	Presenter exhibits some enthusiasm or is not heard consistently, but has confidence in the presentation	Presenter has limited enthusiasm and confidence and does not project voice	Presenter has little or no enthusiasm, lacks confidence, and mumbles
Posture and Presence	Presenter maintains good posture and makes eye contact with the audience.	Presenter maintains less than good posture or makes limited eye contact.	Presenter leans or slouches and makes limited eye contact.	Presenter is lethargic and makes little eye contact.
Understanding of product	Presenter understands all aspects presented about the project.	Some gaps in understanding when questioned by audience	Some gaps in understanding when looking at team slides	Poor understanding of project.
Engaging information	Engaging information is presented about the project.	Valuable information is presented but there are definite holes given the expected information or information has some boring parts.	Valuable information is less than un-engaging information.	Little valuable information is presented
Coverage of Ember Client-side integration	Shows how the app interacts with the API and fulfills the desired use cases	Partial Coverage.	Little Coverage.	No coverage.
Coverage of API and auth functionality	Describes API endpoints and auth functionality and demos their operation.	Partial coverage.	Little coverage.	No coverage.
Coverage of Apache/OS hardening	Describes efforts to harden apache conf and OS level	Partial coverage.	Little coverage.	No coverage.
Challenges Discussed	Difficult challenges encountered and their solutions are discussed.	Challenges encountered are discussed, but some issues discussing solutions.	Challenges discussed, but not solutions	Challenges and solutions are not discussed.
Demonstration	Deliverable is demonstrated with accompanied explanation of outcomes and success	Deliverable increment is explained but is missing outcomes and success	Deliverable increment demonstration has limited explanation overall	Deliverable increment is shown but the audience is unclear as to what it did and why
Timeliness	Perfectly within time limit, Finishes within a few minutes of the time limit	Goes over by less than a minute or Finishes with a decent amount of time remaining.	Goes over by 1-5 minutes or Finishes with a large amount of time remaining	Forces the instructor to rope them off the stage or Really? Did you say anything?

Additional Materials: Install and User Guides (120points)

Usage Instructions (60pts)	**Meets (13-15 pts)**	**Some Issues (8-12 pts)**	**Does not meet (0-7 pts)**

completeness	Instructions for use of each product feature are described fully from an end-user perspective.	Some incomplete instructions for some product features	Many incomplete instructions for most product features.
clarity	Instructions are easy to understand and follow.	Instructions are somewhat easy to understand and follow	Instructions are not easy to understand and follow
readability	Instructions do not contain jargon that end-users wouldn't understand.	Minimal jargon usage.	High amounts of jargon that end-users would likely not understand.
coverage	Instructions cover all important product features and functionality.	Instructions cover most product features and functionality.	Instructions cover only a small subset of product features or functionality.

Install Instructions (60pts)	**Meets (13-15 pts)**	**Some Issues (8-12 pts)**	**Does not meet (0-7 pts)**
completeness	Instructions for installation are described fully from an admin perspective.	Some incomplete instructions for some install steps	Many incomplete instructions for installation.
clarity	Instructions are easy to understand and follow.	Instructions are somewhat easy to understand and follow	Instructions are not easy to understand and follow
readability	Instructions do not contain jargon that admin users wouldn't understand.	Minimal jargon usage.	High amounts of jargon that admin users would likely not understand.
coverage	Instructions cover all important steps to install and configure the app.	Instructions cover most steps.	Instructions cover only a small subset of steps.