IA-8470
Project 1: Requirements Engineering, Product Design, Architecture, and Client-side Prototype
Assigned: Friday 9/27/16
Due Date: Part 1 check-in Oct 11 (10/11/16) at 11:59 pm
Full submission due Nov 1 (11/1/16) at 5:29pm
Presentations: Nov 1 (11/1/16) during class time

Overview

Project 1 will use the skills you learned in Project 1 and have you create some software engineering documents to help you flesh out an idea and implement the client-side portion of the idea that will use for the rest of the semester. For this, and the rest of the projects in the course, you will be able to **use your own project idea**. You will be expected to satisfy the following rubrics. Your project submission materials should include a document structured as follows:

- Executive Summary and Risk list
  - Three paragraphs
  - Extra paragraph for risk analysis
  - Table of identified risks
- Use cases
  - Show use case diagram
  - List selected use cases and misuse cases
  - Brief discussion
- Architecture
  - Show application level diagram
  - Briefly discuss diagram (1-2 paragraphs) to explain components and your design decisions
- Clientside prototype description
  - Lists IP (or heroku url) where the app is hosted
  - Lists github URL where the code repository is
  - Includes the /dist folder from your ember-cli project as an attachment so I can host and run your app
- [attachments]

See the following rubrics for specific details.
Overall Project 1 **is worth a good portion of your overall grade – so don't sluff off!**
You will submit the Executive Summary, Use case Diagram, and Architecture Diagram in Part 1. This is worth 470 points. Part 2, includes your client-side ember prototype, REST API backend prototype, github commits, and in class presentation.
I will give you feedback at the part 1 check-in deadline that you should incorporate into your part two submission. You will be graded once based on the full submission document.

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 number from .4 to 1 that is based on team evaluations, commit history on github/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 (likely) equal to 1 (full points), if you don't you will get a factor proportional to your participation.

Example:
A member of a team (team member *a*) does nothing. The other two people on the team (team members *b* and *c*) carry team member *a*. The team gets an 'A' grade with 1135pts (93%). Team member *a* gets a participation factor of .4, with a grade of .4 X 1135pts = 454pts (37.2%) while team members *b* and *c* get a factor of 1 and therefore get the full 1135 pts (93%). Takeaway: participate and do your part

Executive Project Summary and Risk Analysis (190 points)

You will be expected to turn in an **executive summary** addressing the questions below as well as a **list of potential risks** that your project might encounter. Follow the prescribed structure below.

Clearly communicate your project idea. You should answer questions such as: What are will your app do? Who will use it and why: i.e. what does your customer base look like? What is the value of your app? What security risks might your application encounter? What types of attackers could benefit by exploiting your application? What attack vectors might they use?

Executive Summary: Be exceptionally clear and to the point. Do not use first person or refer directly to the team, yourself, the course instructor, or the class. Construct *three paragraphs* without subheading or bullets describing (1) *software product goals*, (2) *potential user/customer base*, and (3) *value added for the customer* and a fourth (4) for describing your risk analysis (see below).

(1) Software product goals must define the overarching goals of the product and the scope of functionality your team proposes to do by the end of semester.
(2) The user/customer base paragraph must provide an in-depth examination of who might potentially want to purchase, use, or appreciate your product.
(3) The value added paragraph should describe your product's potential benefit(s) to the identified user and customer base as touch on how your product could potentially benefit society in general.

Overall, the executive summary should be exciting and interesting. It is the first (and likely the only) chance for you to engage your reader and, in a real world setting, would determine if your product gets funded. An executive summary does not need technical detail to describe interesting functionality. The reader should understand what you think are the product's merits, scope, and benefits. You should mention the product by name but not use phrases such as "the team", "the class", "the instructor", "project 2" etc.

Executive Summary (90pts)	**Meets (9-10 pts)**	**Some Issues (6-8 pts)**	**Does not meet (0-5 pts)**
Conformance to 3 paragraph topics (10 points per paragraph)	*Software product goals* are well explained and give an achievable scope. *User/customer* base is well defined and clearly characterized. *Value added* to user, customer, and/or society is strong, direct, and believable.	*Goals* are stated but not very clearly or possibly unachievable. *User/customer* is stated but not explained or clearly characterized. *Value* to customer is stated but could be stronger, more direct, or more believable.	*Goals, user/customer* descriptions, and *value added* are not stated or very poorly explained
Proper writing techniques (10 points per paragraph)	Proper spelling, punctuation, complete sentences, and reasonable length paragraph	Some spelling issues, punctuation errors, incomplete sentences, and/or paragraph is short or overly long	Rampant problems with spelling, punctuation, incomplete sentences, and/or paragraph length
Clear discussion (10 points per paragraph)	Relevant, well-founded points made, that are cohesive and engaging.	Reasonably relevant well-founded points made, and/or a minor lack of cohesiveness or interest	Few relevant, well-founded points made and/or lack of cohesiveness and interest

Risk Analysis: Add a paragraph to your executive summary that identifies the 5 most meaningful risks your team is concerned with. You must (1) generally describe these risks in text (think about attack vectors, impact, etc from a business level) and (2) list more technical information about them in a tabular form following the table format below.

Table Format:
Name/Identifier – (Impact x Probability = Magnitude), Short Description, Attack Vector(s), Mitigation Strategy.

Example:

Identifier and Impact	**Short Description**	**Attack Vector(s)**	**Mitigation Strategy**
Risk of sensitive data leakage (8 x 5 = 40) [Moderate]	Could allow identifiable user information to be leaked to unapproved users.	Incorrect access control implementation, inadequate data separation, unsecure communication methods	Implement least privilege, ensure data seperability by user type, use https, session-based server-side validation, and limit use of cookies

Risk List (100pts)	**Meets (5 pts)**	**Some Issues (3-4 pts)**	**Does not meet (0-2 pts)**
Name – (impact x exposure) (x5)	Unambiguous, meaningful name/id. Agreeable magnitude of impact and exposure.	Meaningful name but could be improved or possible magnitude issues.	Little attempt at meaningful name. Disagreeable or no magnitude.
Description (x5)	Clear, succinct, well stated, and understandable description of risk and relevance.	Somewhat ambiguous or lengthy description of risk and relevance	Little care taken to describe risk, lack of relevance, or not succinct.
Attack Vector (x5)	Clear, succinct characterization of how an attacker might realize the risk.	Ambiguous or overly lengthy attack vector characterization.	Little to no understanding of potential attack vectors.
Mitigation strategy, (x5)	Strategy is clearly stated and could be implemented within the project time and scope	Strategy is stated and might be implementable within the project time and scope	Strategy is poorly stated and cannot be implemented within the project time and scope

Use Case Diagram (150 points)

Describe the main use cases you foresee your application handling (i.e. what will your application do. Use case listings in the diagram must have a verb-noun title, e.g. [for facebook] add friend, view feed, join group, etc and identify an actor, e.g. admin or facebook user. Your use case diagram should indicate relationships, e.g. «extends» or «uses», between the use cases, if they exist. The use case diagram should be created in lucid chart and be shared with the group and myself.

You will be graded by the overall coverage of your identified use case listings and use case diagram and on the specific explanation of at least 5 use cases you highlight from the diagram.

Overall use case diagram	**Meets (19-20 pts)**	**Some issues (14-18 pts)**	**Does not meet (0-13 pts)**
Coverage	Lists a reasonable number of overall use cases for your product	Some obvious missing use cases for your product	Few or no use cases developed for your product.
Diagram	Includes all listed use cases and actors, clear connectivity and labeling given.	Some issues expressing or connecting use cases.	Use case and actors are wrong or muddled. Connectivity not clearly established.
Misuse cases	Use case diagram includes a reasonable number of correctly labeled (alternate color actor and ellipses) misuse cases that map to the identified risks in the executive summary and threaten one or more use cases.	Some issues with labeling, identifying misuse cases or bad actors from your risk list or there are too few.	Many issues identifying, labeling, or describing use cases
Mitigating use cases	Use case diagram includes at least one mitigating measure for identified misuse cases. Mitigating measures are shown as use cases with dotted elipses.	Missing mitigating use cases, or labeling issues.	Poor or no attempt made to identify mitigating use cases.
Stereotypes and line labeling,	Use case diagram demonstrates effective use of UML stereotypes (e.g. «extend», «include», «threatens», «mitigates») and the correct lines are used (e.g. actor inheritance or use case generalization).	Some issues using stereotypes or the correct UML connectives.	Many issues using UML stereotypes or connectives.
Use Cast Listing	**Meets (9-10 pts each)**	**Some issues (6-8 pts each)**	**Does not meet (0-5 pts each)**
Explanation (x5 [pick 5 most important])	Use case is clearly explained using the specified use case text format (i.e., "The use case begins when …) given during lecture 3	A few issues with the expression or format of use case explanation	Poor or non-existent use case explanation

Architecture Diagram (80 points)

Submit an architecture diagram that accurately reflects your product's architecture from an application perspective (module components, connectors, etc). The emphasis should be on conceptually modeling how the application works in a way that could be communicated to a customer or software development team. You may use lucidchart, visio, or a program of your choice to create the diagrams.

Architecture diagrams	**Meets (16-20 pts)**	**Some issues (12-15 pts)**	**Does not meet (0-11 pts)**
Application Architecture Diagram - understandable	Fully understandable and informative representation of known entities at the application level. Components are minimally connected and maximally cohesive.	Mostly understandable and informative	Poor attempt at creating a good, informative diagram
Application Architecture Diagram – component completeness	Deliberate attempt to create a complete architecture representation with all necessary components to realize the use case diagram.	Reasonable attempt.	Too sparse.
Application Architecture Diagram – security component completeness	Deliberate attempt to create a complete architecture representation with all necessary security components to realize the use cases that mitigate the misusecases.	Reasonable attempt.	Too sparse.
Application Architecture Diagram Explanation	Clear paragraph of understandable text that explains the diagram and included components.	Somewhat understandable or some entity descriptions missing.	Explanation sparse or missing.

Part 1 Check-in (50 points)

Checks in with instructor on required check-in due date.

Meeting check-in	**Meets (50 pts)**	**Some issues (12-35 pts)**	**Does not meet (0-11 pts)**
Checkin	Sends instructor an email or slack message by or before part 1 check-in date with the various part 1 required documents.	Email is late or missing elements required for part 1 check-in.	Many missing elements or extreme lateness of check-in email.

Ember Application Prototype Functionality (310 points)

Implement a client-side application in Ember to realize the application vision laid out in the project summary, use case diagrams, and architecture. Your ember application should meet the basic requirements outlined below. You will be graded based on appropriate use of ember concepts and use of secure design practices when interacting with, storing, and displaying data in the user interface.

You are expected to:

(1) Use the ember router to:
   a. Route url requests to the correct controller/template
   b. Load data into controllers, where appropriate for use by the templates
   c. Handle actions that modify routes (where appropriate)

(2) Use ember models to:
   a. Represent data as javascript objects
   b. Utilize the *store* for loading and saving data
   c. STUB OUT data end-points as necessary

(3) Use ember controllers and/or components to:
   a. Manage templates
   b. Execute user actions
   c. Maintain and update data based on user actions

(4) Use ember templates / handlebars to:
   a. Display a dynamically updating user interface (where appropriate)
   b. Issue actions to the controller to execute code (where appropriate)

(5) Use Good Security Practices:
   a. STUB out an authentication component
      i. Use clientside redirects where appropriate
      ii. Use router hooks appropriately
   b. Support good client-side data storage practices
      i. Make appropriate usage of localStorage, sessionStorage, and cookies
   c. Validate and sanitize user inputs appropriately at the clientside
      i. In addition to the server-side protections in your REST API (later projects) it is a good idea to validate input fields prior to submission. This means a user doesn't have to wait for the server to respond if the form fields are wrong – it also reduces bandwidth and server usage consumption
      ii. All handlebars fields should be used appropriately. Un-escaped, raw html (i.e., {{{stuff}}}) should be used sparingly and only where absolutely necessary.

(6) Use bootstrap, other CSS frameworks, or custom CSS to:
   a. Present a visually appealing interface
   b. Provide compatibility with modern browsers (firefox, chrome, IE11, safari).
   c. Be mobile friendly (responsive)

Good Security Practices (120 pts)	**Meets (26-30 pts)**	**Some Issues (16-25 pts)**	**Does not meet (0-15 pts)**
Authentication	Stubs out an authentication mechanism for use with the REST API.	Minor security or logical issues with client-side authentication stub.	No authentication stubbed out.
Client-side data storage practices	Uses local storage and session storage appropriately.	A few issues where a data item is stored in the wrong type. E.g. session data stored in local storage or vice versa.	Many issues using local storage and session storage.
Input validation	Makes good use of clientside validation techniques to validate form inputs.	A few obvious missing client-side validations.	No or little attempt to validate fields.
Escaping HTML Display Fields	Follows good practices for escaping HTML displayed to users. Raw HTML, i.e., {{{stuff}}} is rendered	A few issues with non-escaped HTML where it should be escaped.	Little or no attempt to follow best practices for displaying raw HTML.

	sparingly and only with appropriate rationale		
User interface design and CSS Usage (90pts)	**Meets (26-30 pts)**	**Some Issues (16-25 pts)**	**Does not meet (0-15 pts)**
User Interface Visual Design	The application presents an attractive, functional, user interface design.	Minor usability or visual issues with the user interface design.	Many usability or visual issues complicate the user interface design and distract from the application purpose.
CSS Library Usage	Effective use of CSS Libraries such as Bootrstrap, Bootstrap for ember, Less, Foundation 3, Skeleton, YAML, ETC is evident.	Some issues with using CSS libraries incorrectly causes minor performance issues with CSS usage.	Many issues with CSS library usage incurs a large performance impact for the application.
Responsive	Displays good responsiveness, especially for mobile viewports.	Some issues that cause the application UI to appear poorly on small screens.	Not responsive to view screens.
Application 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.

Moving toward Django API Functionality (100pts)

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.

FOR NOW YOU ARE EXPECTED TO:
Have models mocked out in your API of choice (probably Django) or use ember http-mocks or another API mock app to support basic data requests from your client-side where appropriate. What do I mean by this? I mean you need to have some basic data handling functionality built into your app for Project 1. Your ember front-end should be able to make ajax requests and get data back from your "API." I put it in quotes because it can be mocked-out at this stage.

This basic API (mock or otherwise) is worth 100pts and it will be graded strictly based on the following rubric:

Models (100pts)	**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.
API is available as JSON	JSON fitting your model fields is available at your API endpoint (mock or real) – i.e. a URL	Some API endpoints are functional, others are not	No evidence of functional API endpoints

IN PROJECT 2 YOU WILL BE EXPECTED TO: (will be 540 points in project 2) [NOTE THE FOLLOWING IS FOR REFERENCE ONLY]

(1) Create django models to support your ember app
   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 the admin interface.	Application uses explicit session-based auth as defined in settings.py but sessions are not available in admin interface	Application does not use explicit session-based auth.
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	Most validator use is well founded and applicable, but	Poor or no use of validators where needed

	custom validations (such as email, zipcode, or ssn fileds) to ensure fields are well formed.	some obvious needs are missed.	
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.

Github usage (50 pts)

It is very important that you use git and github THROUGHOUT project 1. 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 1 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
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
Use cases covered	Use cases are covered by the talk – including what are the actors and what are the actions.	Most use cases are covered but some are left unexplained.	Poor use case coverage. Little information about actors and actions.	Use cases not covered
Client-side app and architecture diagrams covered	App is shown in conjunction with architecture diagrams and a discussion of why the architectural design was chosen.	Some discontinuity between app and architecture diagrams. But coverage is good.	App not covered with architecture diagrams.	Architecture diagrams and design reasoning not covered.
Risk analysis	Clear discussion of potential risks.	Minor issues answering questions about risks.	Risk analysis is very unclear during discussion.	Risk analysis not covered.
Within Time limit	Perfectly within time limit	Goes over by less than a minute	Goes over by 1-5 minutes.	Forces the instructor to rope them off the stage.
Not too short	Finishes within a few minutes of the time limit	Finishes with a decent amount of time remaining.	Finishes with a large amount of time remaining	Really? Did you even say anything?