ELEC3609/ELEC9609 – Project Specification

Group Project worth 60% of overall grade.

2017 – v1.1

This document is subject to changes.

Dr. Peter Phillips

peter@freelancer.com

Lecturer

Introduction

“**Software is eating the world**”

– *Marc Lowell Andreessen*

Web applications are ubiquitous and their popularity is ever-increasing. What was once a brick-and-mortar store or specialised software package now operates exclusively from a web browser.

Whilst seemingly simple, web applications cleverly hide a wealth of complexity and human-effort from their users. This project will explode these hidden complexities to familiarise you with the concepts and practices which form the backbone of today’s leading web applications.

You and your team-mates will conceive an idea for a web application. You will take this idea through a design phase, where your team will express and clarify your vision through formal modelling, analysis and specification. Your team will work together to realise your designs by coding the user interface and server-side logic in an advanced web framework of your choice. Finally, your team will host your custom software piece in an enterprise cloud environment and present your running, world-facing web application to your peers.

Timeline

Assessment	**% mark**	**Due (wk)**	**Due (date)**
System Requirements Analysis & Use Case Modelling	10.00%	5	31/08/2017 5:00PM
System Design Specification	5.00%	7	15/09/2017 11:59PM
System Implementation	30.00%	12	27/10/2017 11:59PM
System Deployment & Testing	15.00%	13	02/11/2017 5:00PM

The Web Application

It is up to your team to devise an idea for your web application. It is not a requirement that your application is unique or commercially viable. The idea must adhere to all relevant laws and be in good taste. Be aware of governments, commercial enterprises or interest groups which may take issue with your application - this is a project to learn about internet platforms, not push legal or social boundaries.

Typically, your application should include a few main functions to efficiently address a broad problem statement. For example, if your problem statement is 'Existing time-sheeting solutions are too bloated and waste time', your application should implement time-efficient functions to create, update and review time-sheets.

In order to ensure this project provides students with experience in critical web technologies, the items in the following list must appear in your web application.

User Accounts and Authentication

Your application must implement a persistent store of users and allow them to authenticate via HTTP(S). This involves providing functions for the following:

- User creation and/or enrolment.
- Authentication via username and password, or token.
- Session management via cookies.
- At least one function or view which is available only to authenticated users.

AJAX

Your application must serve a HTML document which, after loading, performs an asynchronous request to your application's back-end.
Some information from the request must be displayed to the user via an update to the DOM.

RESTful API

Your application must provide at least one API endpoint supporting the JSON format.
The endpoint must allow API consumers to interact with your database and perform create, read, update and delete actions using appropriate HTTP verbs (i.e. PUT, GET, POST, DELETE).

CSS

Your application must, at a minimum, serve a HTML file which implements a separate CSS file written by your team. This CSS file must cause a visible change to the rendering of the website.

If these items don't make sense for the core features of your web application, simply add a small feature which implements them. For example, you can add a weather feed to satisfy the AJAX requirement.

You application must provide it's primary user interface to users in the form of HTML, CSS and JS documents served to a web browser.
This does not include Android, iOS or native desktop apps.
Broad cross-browser compatibility is not within the scope of this project, you are only required to provide compatibility with Google Chrome on Windows and/or OSX.

System Requirements Analysis & Use Case Modelling

Due: Week 5 – 31/08/2017 5:00PM

The system requirements analysis phase is critical to the success of any non-trivial software system.

In a commercial project, this will provide the foundation of the contractual engagement between investors and developers. In any project, this phase provides direction to stakeholders, illuminates potential issues, identifies requirements, and defines outcomes.

Software Requirements Specification Document:

The purpose of the SRS is to ensure all persons involved in the project understand exactly what the project requires.
One way to think about it is: could a person who has had no contact with your group pick-up your SRS and build the project for you?
The tutors will use this document when marking your completed project in order to determine if you did what you said you would do.
That said do not limit yourself based on this fact. If you wish to include extra features, be sure to differentiate between those in the MVP and those that are 'bonus'.
You will be required to create a document which incorporates the following components to prove you have satisfactorily planned and designed your future web application:

Website wireframes

Wireframes are used to define the basic layout and structure of a specific web page or view. Effective wireframes omit colours and theming to focus exclusively on UI flow, features and layout. Your requirements document is to feature wireframes for each generic view of your application.

Use Cases

Your document must include use case requirements. You must document all expected use cases for your application and at least one invalid or malicious use case. You must formally define uses cases in English statements and supplement them with diagrams. You must communicate your use cases clearly.

UML (Unified Modelling Language) is the expected language for use case diagrams, however you are not required to adhere strictly to the UML specifications. Diagrams must not introduce new concepts and must only be used as a supplementary communication tool for concepts already formally defined in plain English.

Project Plan

Your requirements document will include a project plan covering the following aspects at a minimum:

- Assignment of responsibilities to team members.
  For example, who will write the AJAX JavaScript and who will be responsible for the database.
- Milestones with dates to track your team's progress.
- Identify MVP (minimum viable product) features which must be present and other features which may be abandoned for the Week 13 presentation in case of time/resource restraints.

Presentation

You will be required to present your idea to the class in a five minute presentation. The five minute time limit will be enforced.

System Requirements Analysis & Use Case Modelling continued on next page. . .

This presentation should emulate a short pitch to investors, describing only the core concepts and innovations of your website. Your intention is to impress your audience with the merits and potential of your future web application.

You will be marked on how effectively you can communicate how your application addresses the needs of your users.

It is not compulsory for all project members to speak during the presentation, but they should all be prepared to answer questions afterwards.

Marking Criteria

	Novice	**Competent**	**Proficient**
Wireframes	**0:** Not done, or done poorly.	**1:** A reasonable attempt, but missing key pages or too much graphical detail.	**2:** Comprehensive coverage, appropriate level of detail.
Use Cases	**0:** Not done, or done poorly.	**1-2:** Not enough use cases or some use cases are ambiguous, unclear, or inconcise.	**3:** An appropriate number of concise and unambiguous use cases which can be easily measured, with an appropriate diagram.
Project Plan	**0:** Not done, or done poorly.	**0.5:** Unachievable timeline or inappropriate delegation of tasks.	**1:** A sensible and achievable plan.
Document Formatting	**0:** Unprofessional and poorly structured.	**1:** Satisfactory, but not very professional.	**2:** A professional document that could be presented to clients.
Presentation	**0:** Did not present, or presentation was very unclear.	**1:** Some components presented clearly, but a generally unprofessional presentation.	**2:** A professional presentation that could be given directly to clients.

System Design Specification

Due: Week 7 – 15/09/2017 11:59PM

A system design specification (SDS) is an internal document (not client-facing) which acts as the oracle of information required to develop the system.
The SDS will not only provide value as a reference, the process of producing it will help your team understand in much more detail the challenges involved in your task.

You will be required to produce a system design specification with the following components:

Data Model
Your SDS must provide one or more Entity Relationship Diagrams (ERDs) in Crow's Foot Notation which exhaustively illustrate each table you envisage in your proposed application's database.
In the unlikely case that your database is such that you cannot express it clearly using an ERD, you will be expected to use another more suitable documentation method.

Code Structure

- Style guides.
- File/folder layout

Dependencies
You must document the versions of the third-party software packages your application will leverage. This includes databases, libraries and programming languages.

Exclusions
What won't your app do? Use exclusions to further define your application. Protect your project from scope creep by excluding features which may be expected by stakeholders but do not fit in with your vision, resources, or timeline.

Marking Criteria

	Novice	**Competent**	**Proficient**
Data Model	**0:** Too simple, missing important features, or poorly normalised.	**1-2:** Missing minor features, could be normalised further, and/or unclear presentation.	**3:** Full feature coverage, fully normalised, presented clearly.
Code Structure	**0:** Not done, or done poorly.	**0.5:** Attempted, but missing important clarifications.	**1:** Structure described with appropriate detail.
Dependencies	**0:** Not done, or done poorly.	**0.5:** Attempted, but missing important dependencies.	**1:** All important dependencies identified and described.

System Implementation

Due: Week 12 – 27/10/2017 11:59PM

By week 12 you will have implemented your application as per your previously submitted SDS.

Code Base

Your code base will reside in one or more git repositories which your tutors and lectures will have access to. These repositories will be hosted on either GitHub or Bitbucket. At submission your repository will be cloned and any code which has been pushed will be considered in your submission.

You must grant access to your repository both programmatically and via the web console. Please use the SSH public key on page 11. Please provide a username & password to access the web console.

Project Management

Your team is required to use either GitHub or Bitbucket project management tools. These should be used as your team's only method of formal communication.
Whist informal communication should occur during your project, expect it to be ignored by markers. Markers will use git commits and GitHub or Bitbucket *Issues*, *Projects* or *Tasks* to assist in determining the level of contribution of each team member. Considering this, each team member is to use exclusively their own GitHub or Bitbucket account.

Presentation

You will demonstrate your application to the class during the week 13 tutorial during a 5 minute presentation. You may go slightly beyond the 5 minutes, but you will be cut-off at 7 minutes.

The presentation should emulate the format of the pitch to investors performed in week 5. You should identify the main features of your presentation, and provide a working demonstration of what you have completed.

If you intend to do a *live* demonstration, be sure to practice it beforehand to ensure everything works as you expect. Technical issues during the presentation will not be ignored by the markers.

If your application is a mobile web application, you should consider how you will effectively demonstrate it to the class. Chrome's Device Mode is a good way to emulate a mobile device on a larger display.

Documentation

At the end of week 12, you will submit documentation to support your completed application. This is reflective documentation which should be written for an audience of other developers.
Your documentation should feature the following components at a minimum:

- **Data Model:** Include the data models provided in your SRS featuring any changes made during the implementation process.
- **Code Structure:** Include the documentation on code structure you provided in your SRS featuring any changes made during the implementation process.

System Implementation continued on next page. . .

- A comparison of your finished project to the SRS
- Describe how your finished project compares to the product you defined in your SRS.
  - Did you fail to fully meet your specification? Did you exceed it? Why?

Your documentation will be assessed from the perspective of another developer, who has been brought in to take over your project. Your documentation should include all details necessary for another developer to get up-to-speed on how your system works, the code layout, and any dependencies or issues that they may face.

Assessment of team dynamics

As a group, assess how your team worked together.
Did you find issues with multiple people working on the same code base? Did you discover processes which improved group efficiency? Did you have lazy people? Did one person do all the work?

After week 13 your team will be provided with an opportunity to assess the *effort* of other team members via an online form. In this form, you will describe the effort expended by your team members in your group attempt to complete the project. Any team containing members who did not exert sufficient effort will have their marks reviewed and possibly reallocated.

Marking Criteria

The overall quality, professionalism, and completeness of your code will be assessed on the whole. Below is a non-comprehensive list of qualities that will be examined:

CSRF Prevention:

Implement a CSRF token to prevent CSRF attacks.
If your framework provides CSRF protection (as Django does) ensure you implement it as documented.

No XSS:

Ensure your HTML is escaped when being displayed.
Your framework should provide a sensible templating language that will help you prevent XSS.

No SQLi:

Your application must not be vulnerable to SQLi.

System Implementation continued on next page. . .

	Novice	**Competent**	**Proficient**
Code Structure & Style	**0-1:** Difficult to read, lots of copy-pasted code, poor program structure and/or folder layout	**2-2.5:** Sensibly laid out, some comments, inheritance where necessary.	**3:** Easy to read, appropriate amount of comments, code is reusable (proper class inheritance, dependencies separated, etc)
Satisfies Requirements	**0-2:** None or very few requirements are satisfied	**3-5:** Major requirements are satisfied but with some mistakes or omissions	**6-8:** All major requirements are satisfied.
Req: User Login/Sessions	**0:** Not Done	**0.5-1:** Used somewhat	**1.5:** Used significantly and appropriately
Req: AJAX	**0:** Not Done	**0.5-1:** Used somewhat	**1.5:** Used significantly and appropriately
Req: REST API	**0:** Not Done	**0.5-2:** Implemented somewhat	**2-2.5:** Used significantly and appropriately
Req: CSS Styling	**0:** Not Done	**0.5-2:** Used somewhat	**2-2.5:** Used significantly and appropriately
Security Checks	**0:** Significantly vulnerable	**0.5-2:** Some vulnerabilities	**2-3:** No significant vulnerabilities
Documentation	**0-1:** Difficult to follow, missing important information.	**2-2.5:** Document covers important concepts, with only minor omissions.	**3:** All important concepts are covered fully, document is of a professional standard.
Presentation	**0-1:** Presentation is difficult to understand or appears to be significantly under-prepared	**2-3:** Presentation makes sense, communicates the basic concept, demonstrates teamwork	**4-5:** Presentation is clear and concise, shows preparation, and appropriately covers the important aspect of the project.

System Deployment & Testing

Due: Week 13 – 02/11/2017 5:00PM

By week 13 your application will be running in a production environment in “the cloud” and your application will have at least one functional unit-test.

Deployment Environment Specifications

Each group will setup access to Amazon Web Services provided by AWS Educate. You must create a Linux instance on AWS which will host your website. Do not expect a graphical environment - you will need to configure this server from the command line. As your team will be required to implement your web application on this server, you should ensure your web application can run under following specifications (these are of an AWS t2.micro instance).

- Ubuntu Server 14.04 LTS
- 1 Virtual CPU (Intel Xeon)
- 1 GiB RAM
- 8 GiB Storage (EBS)

You are free to create any type of Linux instance on AWS, however be weary of how much free credit you will be allocated. If you exceed the free credit available, you will need to pay for the services yourself.

Deployment Requirements

You will need to implement the following requirements in your production environment.

- **Reverse Proxy (nginx):** Requests from the internet to your web application should hit a reverse proxy, which will forward appropriate packets to your web application.
  You must reduce server load by configuring nginx to serve static files (i.e. JS, CSS, images, media, etc.) instead executing your application code.
- **Fault Tolerance:** If your server reboots, it should not need human intervention to begin serving requests as normal again.

Unit-testing

Provide unit tests for at least five different pieces of functionality, making a reasonable attempt at covering a significant portion of your web application’s back-end.
If your framework has support for unit-testing (as Django does), implement it in your frameworks style.

Security

Your server, database and AWS account must be configured with appropriate security precautions and restrictions in place.

Documentation

You must provide some developer documentation to describe how the deployment environment is constructed. This should aid any future developers in uncovering how your system operates.

You may incorporate this documentation into your final System Delivery documentation.

System Deployment & Testing continued on next page. . .

Access for Markers

You must provide markers access to your AWS interface and your Linux server instance.
To do this, you *must* configure the following components:

- **IAM User**
  An IAM user must be configured with 'ReadOnlyAccess' to your AWS account. This will allow the markers to access and assess your AWS set-up. Include the 'IAM Users Sign-In Link' and credentials in your Week 13 documentation.

  Following are basic instructions on creating an IAM user with 'ReadOnlyAccess':

  1. Log into the AWS Management Console.
  2. Open the 'Identity and Access Management' (IAM) service.
  3. Take note of the 'IAM Users Sign-in Link'.
  4. Under 'Details', click 'Users'.
  5. Click 'Create New Users'
  6. In the '1.' field, enter a username for the marking account.
  7. Click Create.
  8. Click Close (there is no need to download the credentials).
  9. Back in the IAM section, in Users click on the new user you created.
  10. Select the 'Security Credentials' tab.
  11. Click 'Manage Password'.
  12. Select 'Assign a Custom Password'.
  13. Enter and confirm a new password for marking access.
  14. Click 'Apply'.
  15. Select the 'Permissions' tab.
  16. Click 'Attach Policy'.
  17. Select the 'ReadOnlyAccess' checkbox.
  18. Click 'Attach Policy'
  19. Test the account: browse to the 'IAM Users Sign-in Link' and ensure the new user has read-only access to all aspects of your web application.

- **SSH Access**
  You must append the following SSH public key to the file: `/home/ubuntu/.ssh/authorized_keys`. This will allow markers to log into your Linux instance and assess it's configuration.

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC96fmaBKrsFurlr39PxdUhDze4M43a
9WzEGPPBBgMmm48WmvujIvSyqrS/qMpaOz+wPSFSduSiA47X/anjG0xFRrLcBVCLrlvmmFxtml73kyGeTUQ
EUYKbMXEi+TbfoXkUl+oDuj973UUpUIqyJzUSZArKKwZWZH1iqLMlOajeLu55vZTxnmgREFt96CxZxCNau1
IWkMe0Y1QPx75RLJnvqX5u2sIV4l0p9plMq7z3WHwPKvp673tM0nBzNGsDTKSGbu5EjpaQWuZ5pW7Vb66Cp
LeUk1047xr5T2+uF1uQnbfjxgZEPBYtrt0sHHbb1cWrGRfJQ2zFXsUOaVEJKtrLHnr4gYqtdQPGUYwaa490
TB0DjC65xmY8aMT6PHrWvLvZhd7SSten7c7nEtduGvLQXL/P8xRlB4FYVAtgt1jH2R64sLC6wPeXu2x4mrK/
1w802cgY3l3gZTPT2SnHmab3+vWVF3a471GphD4jdWjJX+ITv3v9J/gdFTgfZKDKqgJzX6d3K982+RjVftBD
Tu/us8HUME4cXw6cIiJl6rYAXYeMJkvId+op2hXWv12uCUCXw/RR03Y85TbQ1akgJbN943ARiJvTuZc4kJ9U
MfG6bHuByZJZCJ2zNczejlb4myh9LXakFrSjMZGz6T2Co8gcfchwatNFrVAqhCP0B9T+bw== markers@elecx609.com
```

  (ensure that the line-breaks are removed and that this all appears on a single line)

- **Database Access**
  You must ensure your application server has a functioning terminal-based frontend to your database (eg, `psql` or `mysql`) to allow markers to review your database state and structure.

In your documentation, provide credentials and basic instructions for markers to obtain terminal-based access to your database from your application server.

Deploying to production always takes longer than expected.
Do not leave deployment until Week 13 - you must have your production environment running (in a basic capacity) several weeks prior.

Marking Criteria

	Novice	**Competent**	**Proficient**
System Set-up	**0-1:** Instructions have not been followed, server is poorly configured or does not work as it should.	**2-4:** Most aspects of the setup have been followed, but there is room for improvement or some configurations are incorrect.	**5-6:** All specified configurations have been incorporated and the student has clearly invested effort into configuring their server properly and securely.
Unit Tests	**0-2:** Insufficient unit testing or unit tests do not cover enough features.	**3-4:** Unit tests are decent, but don't add considerable value or don't test the most important features.	**5:** Sufficient unit testing with appropriate coverage of website features.
Security	**0-1:** Significant vulnerabilities found.	**2-3:** Minor security issues identified, but would not cause full system compromise.	**4:** No security issues identified, system appears to be secure.

Submission Details

Overall

- **Submissions are due before the time and day specified**.
- Late submissions will be penalised 20% per day late, including weekends.
- Plagiarism will not be tolerated and your assignment may be submitted to a plagiarism checking service.

System Requirements Analysis (Week 5) and System Design Specification (Week 7)

- Please submit these reports in the "Assignment" section of the unit's e-learning site before the deadline.
- Submissions should be in PDF format.

System Implementation (Week 12)

- Documentation must be submitted via e-learning before the deadline.
- All code must be pushed to the group's repository. The most recent commit before the deadline will be marked, but none after.

System Deployment and Testing (Week 13)

- Your documentation must be submitted via e-learning before the deadline.
- The deployment environment should be demonstrated to your tutor during your allocated tutorial.