CSC 648/848 Software Engineering – Spring 2021
Milestone 2

More Detailed Requirements, Specs, Architecture, UI mock-ups and Vertical SW prototype

Henry Villar, CEO & CTO, SFSU
jvillar@mail.sfsu.edu

Acknowledge
Content based on class material created by Dr. D. Petkovic

Objective

Goals of Milestone 2 are:

- ***More detailed design - Milestone 2 document***: prioritize requirements; design high level UI mockups and storyboards; design high level DB organization, and practice UML.
- ***Vertical SW Prototype***: Develop first simple bare-bones prototype (from DB to browser) to test the infrastructure, educate the team, resolve technical issues and also serve as basis of further development (must ruin on the deployment server).

Milestone 2 delivery hence consists of two parts *with separate deadlines:*

- **Milestone 2 document** (one per team, submitted similarly like Milestone 1)
- **Vertical SW prototype** (one per team) to be reviewed in class and also by Class CTO (off-line)

Milestone 2 has to be reasonably consistent with Milestone 1 and instructors' feedback but it can also differ from Milestone 1 based on what you discover and develop in your design process in the spirit of iterative SE process and based on the feedback you get.

Milestone 2 differences DO NOT need to be edited in Milestone 1 doc which remains frozen. You should start with Milestone 2 only after you have incorporated instructors' feedback on Milestone 1. Milestone 2 document is a separate document from Milestone 1 document.

Store all milestone documents in github folder “Milestones/M#”

Milestone 2 Document

Use as much space as you need, but the expected length is about 20-25 pages. You must submit all sections.

The sections you must have in Milestone 2 document are as follows, in the order below::

Title page (see below), followed by

1. Functional Requirements - prioritized

Expand or repeat functional requirements from Milestone 1 into Milestone 2, with more details only if necessary. Keep the same reference numbers with respect to Milestone 1 (i.e. if high level requirement was number 3 in Milestone 1, then Milestone 2 more detailed requirements of requirement 3 are 3.1, 3.2 etc.). *Be sure to cover ALL and especially unique features of your product*. OK to add new or delete previous functional requirements from Milestone 1, if you can justify it.

Prioritize each requirement/spec with 1, 2, 3. (1-*must have*; 2 – *desired*; 3 – *opportunistic* as defined in the class). To develop these priorities, think of the user, use cases, and making your application complete from usability, marketing and business aspects. Base this also on your skills, resources and schedules. Instructors will check final priorities. The priorities you set later in Milestone 3 and 4 will constitute your commitment (especially priorities of 1).

In terms of presentation, for easier review, please group all requirements first by priority i.e. list Priority e.g. Priority 1 requirements first, then Priority 2 etc. and within each priority section you should group them by actors (users, admin)

2. UI Mockups and Storyboards (high level only)

- Create mockups/storyboards for all major use cases from M1 (e.g. 4-5 major use cases, each storyboard appr. 1-3 screens). Have only one to two mockups per page so we can easily read it and comment. Start with black and white wire diagrams focusing on basic layout and description of the functions in each main area of the GUI. Create simple “storyboards” (sequence of mockups) organized by use cases. This helps test the navigation and flow. For each storyboard repeat a short version of related use case so the reader knows what is being done

- The format for UI mockups is very flexible but we recommend hand drawings, which you can scan and include in final Milestone 2 document. Do not use graphics or colors yet (unless absolutely necessary), it draws attention from basic UI concepts (functions, behaviors, layouts, flow…). You can also use some tools like Figma
- Before submitting "Test" the above mockups, keeping ease of use and your use cases in mind. Walk through your mockups following each major use case and make sure your storyboards work well.
- Use data terms and names consistently with Data Dictionary and use cases.
- **Make sure that the actual display of mockups in the hard copy Milestone 2 document is easy to read with max two mockups per page and that they are organized by major use cases**

We recommend front end team be assigned to this task.

3. High level Architecture, Database Organization

- *DB organization*: Describe the main database schema/organization (high level), e.g. list main DB tables (e.g. their titles) and items in each DB table (check instructors' suggestions and class slides on architecture). Make sure the titles and var. names are in easy to understand plain English and consistent with data definitions in Section 1 above.
- *Media storage*: Decide if images and video/audio will be kept in file systems or in DB BLOBs (decision on file vs. BLOBs must be made by the end of M2). Describe any other special data format requirements like for video/audio/GPS etc.
- *Search/filter architecture and implementation*: what will be the alg/SW for search; how will you organize search items for the user; what DB terms will be searched, how it will be coded and organized in the DB (check instructors' suggestions in the class. OK to use SQL and %like).
- Your own APIs (if any): Describe and define at high level any major APIs that you will create other than standard ones provided by tools and frameworks you use
- Describe any significant non-trivial algorithm or process if any (like rating, ranking, automatic prioritizing of items etc.)
- If you have changed SW tools and frameworks or added any new one please describe it. Any new SW or framework you will be using has to be approved by CTO in writing by this time.

4. High Level UML Diagrams

Familiarize yourself with Unified Modeling Language (UML). Find your favorite UML tutorials from the Internet. One good one is
http://edn.embarcadero.com/article/31863

For Milestone 2 provide only:

a*) High-level UML class diagrams* for implementation classes of core functionality, i.e. functionality with provided interfaces. Focus on a main high-level classes only (one or at most two levels deep). This must reflect an OO approach to implementing your site.

b) *UML Component and deployment diagrams*

Use data terms and names **consistently** with Data Definition Section 1 above.

5 .Identify *actual* key risks for your project at this time

Identify *only actual and specific* risks in your current work such as (list those that apply:

- *skills* risks (do you have the right skills),
- *schedule* risks (can you make it given what you committed and the resources),
- *technical* risks (any technical unknowns to solve),
- *teamwork* risks (any issues related to teamwork);
- *legal/content* risks (can you obtain content/SW you need legally with proper licensing, copyright).

Tell us then how you plan to resolve each actual risk you have. The key is to resolve risks as soon as possible. (Note that we will provide you with basic set of images). Categorizing risk as above helps a lot in managing them. Be brief: identify the risk and explain (2-3 lines), list how you will address this issues' (2-3 lines)

6. Project management

Milestone 2 is a good time to make sure front-end and back-end team operate more independently while also agreeing on common interfaces. This makes the team more efficient. Consult the class slides on project management. It is critical to always assign tasks, and for each task know person in charge and the deadline.

In this section please describe in no more than half a page how you managed and plan to M2 and future tasks and what tools you will use. You must start using Trello or similar tools for task management which offer unified dashboard view of all tasks and status.

Submission of Milestone 2 *Document* for Review

Formatting instructions for M2 document must be followed precisely, as outlined below. Submission must be done by the deadline specified; any extension has to be approved ahead of time.

In creating, editing and finalizing Milestone 2 document follow similar team process as outlined for Milestone 1 document. You can use any tool of your choice for creation and managing the M2 document (e.g. google drive) but you are required to put the final submission version of M2 document in github in "milestone" folder. (Make sure to save also DOC version so you can edit it later). This way you will have code and documentation in one place – good for easier access and as your portfolio.

The whole student team submits one milestone document for Milestones 2, as follows (same as M1 submission): Team leads will send e-mail with a link (NOT the attached file) pointing directly to Milestone 2 Document to Prof. Villar jvillar@mail.sfsu.edu with the subject line as specified below. This link MUST point directly to M2 file in the team group directory "Milestones: on Github.

e-mail subject line: Must be "CSC648-848 Spring 2021 Section M Milestone2 Document Team N" in the subject line (M is section 01 or 02, N is a team number (01, 02 etc.).

e-mail body contains **direct link** to Milestone 2 document in team github . File name of the M2 document to which the link is pointing to MUST be: **CSC648-848 Spring 2021 Section M Milestone2 Team*N*.PDF** (*N* is your team number). File format is PDF.

- **First page of Milestone 2 document must include**
  –"SW Engineering CSC648/848 Spring 2021 Section M"
  Project/application title and name (you can use the name you chose for your application)
  –Team number and name – make it clearly displayed for easy reference
  –Names of students (team lead first) -
  Name of team lead and his/her e-mail
  –"Milestone 2"
  –Date
  –History table (as in M1 – two key items: date submitted for review, date revised after feedback)

- **The rest of the document** must have numbered sections outlined above in "Content and structure for **Milestone 2 document** for review by instructors". Each section must start on a new page

Instructor's Feedback, and Freezing the Milestone 2 *Document* for Final Project Delivery

***** All submission instructions must be followed precisely or negative points will be given, Negative points will also be given for any missing section of M2 document ******

After delivery of the Milestone 2 document, you will get feedback from the instructors (Villar) by e-mail, similar as for Milestone 1. This feedback must be used to revise your Milestone 2 and used subsequently for the rest of the project. Please enter the revision summary in history table. (This is similar to Milestone 1 review process).

After this revision you will freeze the Milestone 2 document, place it in github "Milestones/M2" folder and use it for final project document delivery.

Vertical SW Prototype (VP)

In addition to the **Milestone 2 document,** (and working in parallel) the team will create a "**vertical SW prototype**" to test the infrastructure and chosen frameworks and to jumpstart the coding effort. We recommend that the VP be developed by back end team

The purpose of vertical prototype is to early and quickly test basic SW components and deployment infrastructure and frameworks as well as the key architecture patterns and thus to serve as a basic "scaffolding" for final product. It also serves as "teaching and training" tool to bring the rest of the team up to speed on SW, frameworks etc.

The vertical prototype is the form of a code that exercises full deployment stack from browser (with simple *test home page*), via middleware, to DB and back, using only your chosen and approved frameworks and SW components. It has to be deployed from team account on your chosen deployment server, the same way the final product will be deployed. In github we recommend you have a branch dedicated for this.
We recommend that back-end team be assigned the task of constructing this vertical prototype, with front end team helping with front WWW page.

Vertical prototype shall allow one to enter a search term on ***test home page*** (simple home page used to test vertical prototype), then get a response form the DB and render it back on the browser in a simple ***test result page*.**

- UI for *the test home page* can be a simple one containing
    - Class, section and team identification

    - One **free text** entry field (this text is then used in SQL %LIKE search – check architecture slides)
    - One **pull down** for 2-3 search parameter (like main search category like student type, major, demographics, etc.) – this is exercised using simple SQL filter – see architecture slides
- *The test result page* needs to display search results including images and later their thumbnails as well as profile videos (if implemented) in *any reasonable layout* (the goal here is to make sure you can access and display items and NOT the ultimate UI). Thumbnails need to be computed automatically using some open source thumbnail generation SW – you can do thumbnails later after you figure out the basic way to show image
- The DB can have only a few items. The items in the DB shall be encoded with full schema as it is defined by now in M2 document
- Make sure that if user hits "search" or related button, and does not enter any parameters, you display full list of items from the DB

You must use only selected tools and frameworks for vertical prototype and deploy it on your chosen deployment server.

Vertical prototype serves also to help the rest of the team get "on the same page" in terms of SW development. Back end team should also document vertical prototype code well and use it to educate the rest of the team on how to develop the rest of the product. Front end team can use the test home page to establish rules for CSS and UI development. Back end and front-end teams should also agree on common way to connect UI with back end and document it for all.

Some resources related to VP SW arch. and development

- Tutorial with nodejs, developed by our former student and TA Nicholas Stepanov
    - https://medium.com/@nicholasstepanov/search-your-server-side-mysql-database-from-node-js-website-400cd68049fa
- Tutorial with flask, developed by our SE instructor Jose Ortiz
    - https://medium.com/@joseortizcosta/search-utility-with-flask-and-mysql-60bb8ee83dad
- Tutorial with PHP, by Jose Ortiz
    - https://medium.com/@joseortizcosta/search-utility-with-php-and-mysql-as-backend-server-technologies-d3dac5128d8
- How to use and leverage this:
    - Study code to learn
    - Customize for your app; deeply and test then put on master branch
    - Document well, establish good APIs
    - Use as templates/architecture to guide each team member

Also, perform constant code reviews to ensure people follow the templates and APIs

Organizing the DB for VP and beyond

It is NOT a requirement to have your database and application on different servers. as in industry ➔ make it simple

Main options

- Main production DB and local DB for each member
  - Each member deploys and maintains their own localhost database (e.g. on their laptops) and the team has one main production DB for your application.
  - However, keeping these databases consistent is going to be a challenging task
- Main DB and Test DB on deployment server, team accesses them
  - Run 2 databases (under the same DBMS software) on the remote server: production DB and a Test db.
  - Both should be maintained by the same team member.
  - The test and production database should be a similar as possible.

Vertical SW Prototype delivery/submission (may be different deadline from M2 document)

Your team will submit vertical prototype similarly as M0, via e-mail to class CTO and Prof. Villar by the deadline (may be different one from M2 document deadline). The submission format and process **must be followed precisely**, as always. Submission must be done by the deadline specified; any extension has to be approved 24 h ahead of time.

e-mail subject line: Must be "CSC648-848 Spring 2021 Section M Milestone2 Vertical Prototype Team N" in the subject line (M is section, N is a team number (01, 02 etc.).

e-mail body contains the form with your data, **as in Appendix I below**.

Vertical prototype shall be evaluated (but not graded) by class CTO based on:

- Functionality and correct search and results display (be sure to test before sending it)
- Code organization and architecture
- Proper use of frameworks

- Correct deployment on a chosen team server for final delivery

Instructors' feedback on vertical prototype and your responsibilities

Vertical SW prototype will be reviewed off-line by class CTO and you will get the feedback which you must analyze and incorporate as necessary. Then you can use this code as a basis for developing the final app.

We will not grade vertical prototype but you must follow up feedback from instructors and revise accordingly after submission.

However, any issues like sever bugs that prevent evaluation, or incorrect submission process will be considered incomplete delivery and be noted under team grade rubric of SE Process.

Appendix I – form to be filled out and included in e-mail body for M2 vertical prototype delivery

Milestone 2 Vertical Prototype Form
Team Number <N>

The below form is used for submission of needed information for the current milestone. The below table is used to help access certain parts of your web application. Please make sure the information submitted is accurate and up to date to ensure grading of the
milestone is completed in an efficient manner.

Item	Credentials
Website URL	
Website URL to search page	
SSH URL	
SSH Username	
SSH Password/Key	
Database URL	
Database Username	
Database Password	
Link to GitHub page that performs the Search	

Addition information, anything to help with executing or grading of the current milestone
Please explain what is working for your prototype and what can be tested. Also, what search terms to use. Please make sure to explain how to use certain pieces of the log in information. Especially if its more than a simple SSH command. If you server requires a Key please make sure to include it with the submission of the document, regardless if it was submitted with prior milestones. If you are worried about sending keys though emails , you can use links to google drive or drobox etc. Once key has been retrieved you can delete the link.