TOYOTA

Toyota Motor North America Executive Scrum Board

Team 2 --- Scrum of the Earth

Toyota Project Customer: Troy Morgan

University of Kentucky --- CS 499 --- Fall ’17

October 1, 2017

Authors:

Logan Hickey – editor and author of test cases, review and metrics

Irsyad Hanif – author of class diagram and design patterns

Thomas Wheeler III – author of high level architecture diagram

Jonah Starling – author of UI design and description

High Level Architecture

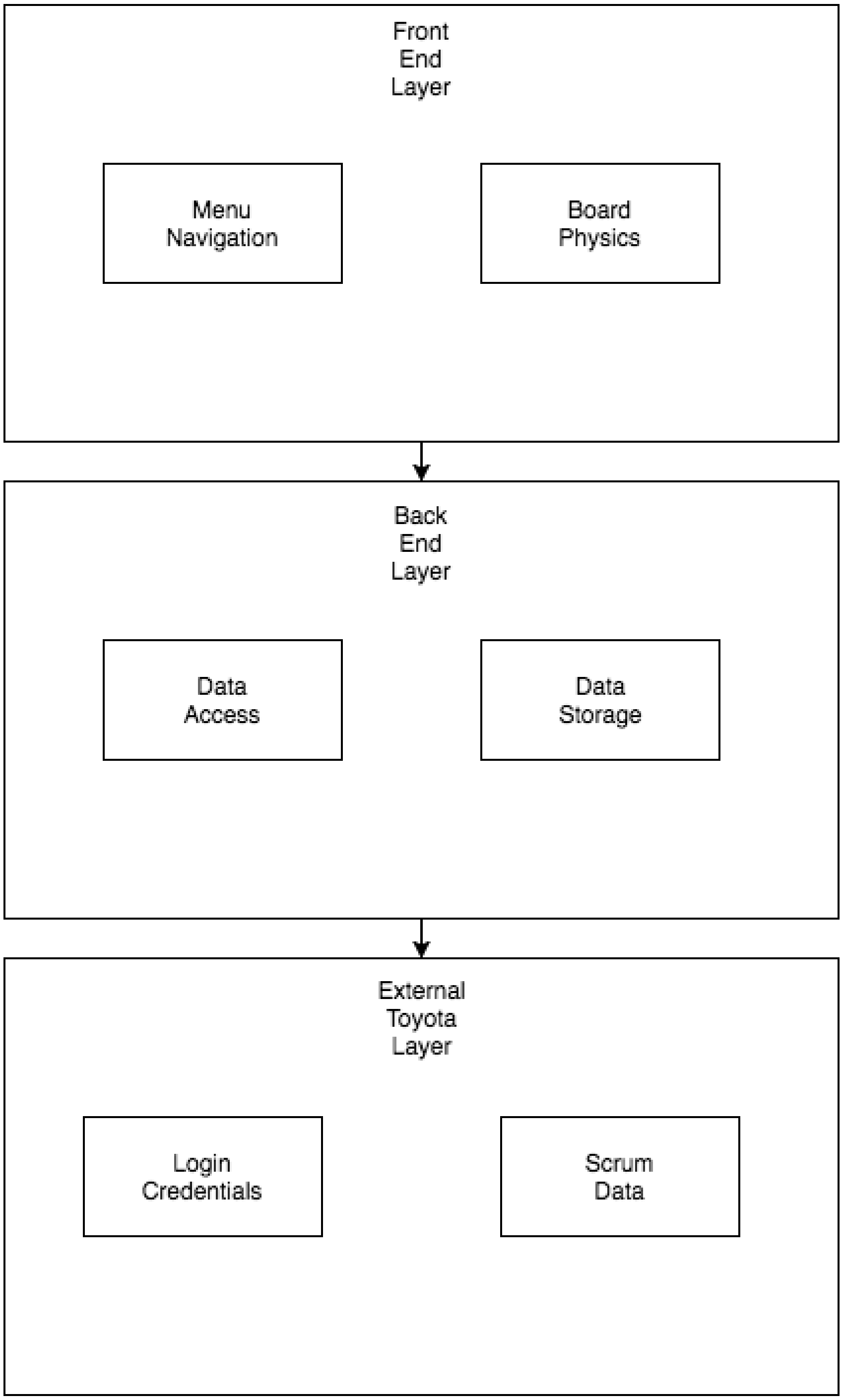


Figure 1. Our architecture model outlines the high-level design for the entire scope of our project, including all of its users. The following section outlines the details of this diagram and their relevance.

The executive scrum tool we are creating for Toyota has 3 major layers. The front-end layer encompasses most of our user interface. The back-end layer handles most of the data access and storage needs. The external layer interacts with existing Toyota databases for user credentials and scrum data.

The front-end layer consists of the menu navigation and the board physics. This encompasses all the interactions a user would have with the scrum board after logging in. The user can use the menu to navigate between boards in different regions. The user can also access setting for the boards and their own user settings from the menu navigations. The board physics involve moving cards around on the board, and opening the expanded card view to show the details of the project. The front-end layer interacts with the back end layer in several instances.

The second layer, the back end, involves all the data and manipulations that we would need to perform to maintain the boards. Data can be imported from other Toyota databases and is stored here. It can then be accessed and viewed as cards on the scrum status board. As information is viewed and manipulated it is updated in real time in the database. As mentioned previously, the back-end layer interacts with several existing Toyota databases including those related to user credentials and those containing pertinent scrum information.

The third layer of the scrum board tool involves accessing external Toyota databases. Specifically, we need to access databases related to user credentials so that users can log in to the scrum tool. We also need to access and store information related to the progress of ongoing projects. We use this data to build and display the cards on each board. Depending on the information, the cards can order themselves in one of the three columns depending on the status of the project.

Detailed Design

Detailed Class Diagram

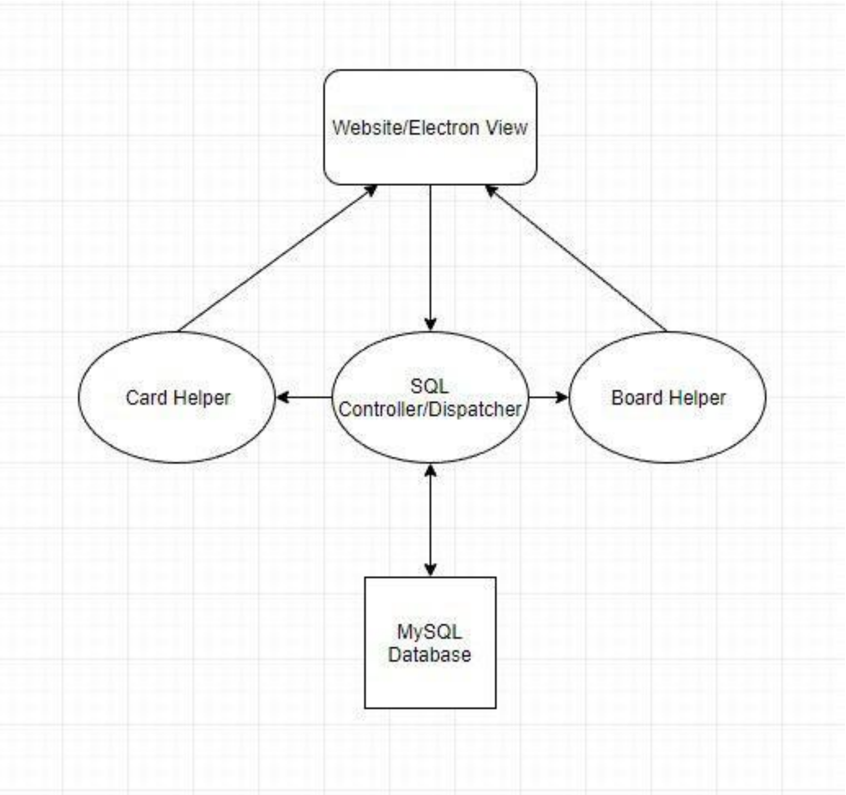


Figure 2. The detailed class diagram is a backend interpretation of the classes that will exist in our projects dynamic database. It shows how these classes will connect and link to the UI. The following section describes these classes that are present in the diagram.

Card Helper Class

This class will hold all the details of cards retrieved from the SQL database. It will have functions to add, delete, and edit them. It will also have a function to update the card in the view class. We are also considering caching the heavy details to secondary storage, and only leaving certain info in memory to reduce memory load.

Board Helper Class

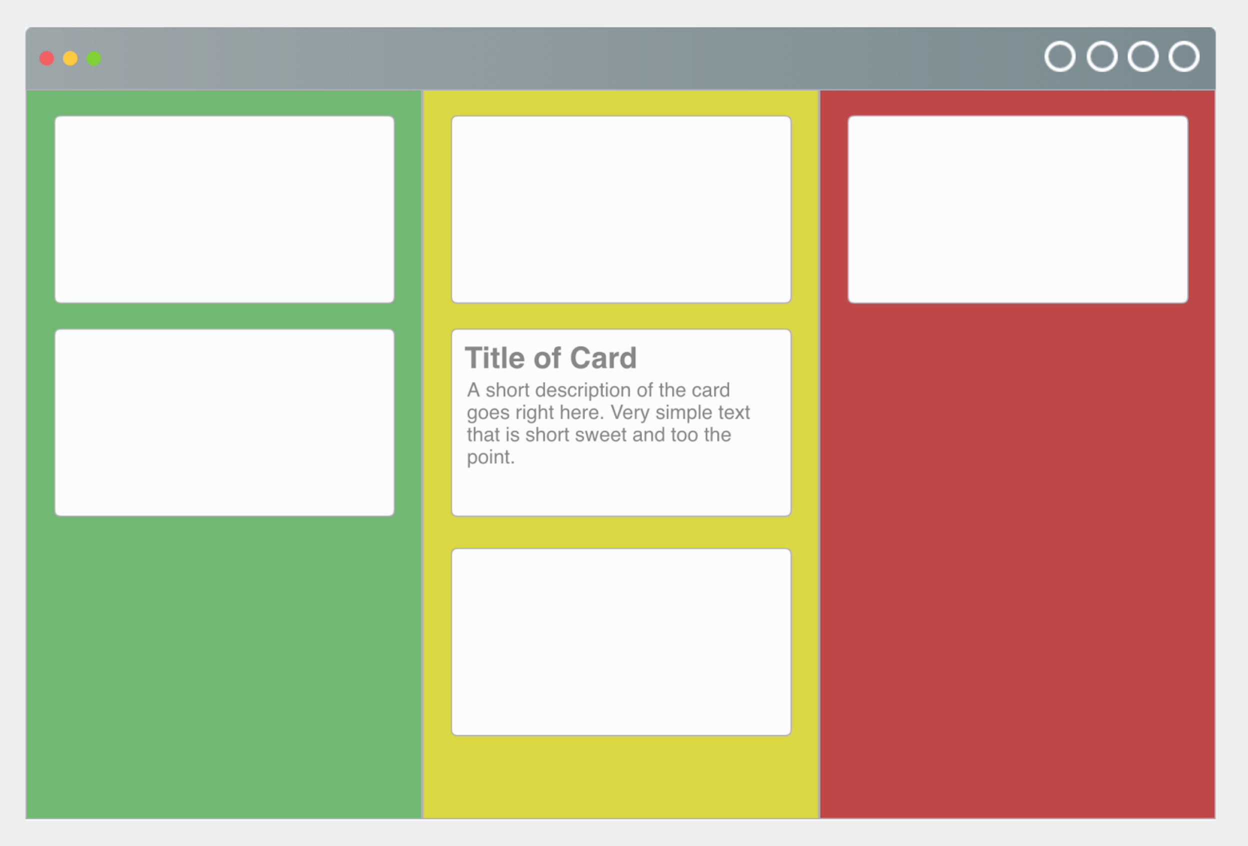
This class will contain the details of the board, the owner, and the array of cards located on the board. It will have functions to get all the cards, set all the cards, and update the board. A user can be part of multiple boards, so we will have to make a distinction between active boards in memory and inactive boards cached in secondary storage.

SQL Controller/Dispatcher Class

This class will interact between both the cards and the board and the user's input. When the user opens a board, or modifies or adds a card, the controller will make the required SQL queries to obtain the desired effect. The controller will receive a query and use it to change the existing objects. These objects then update the existing views on the program.

User Interface Design

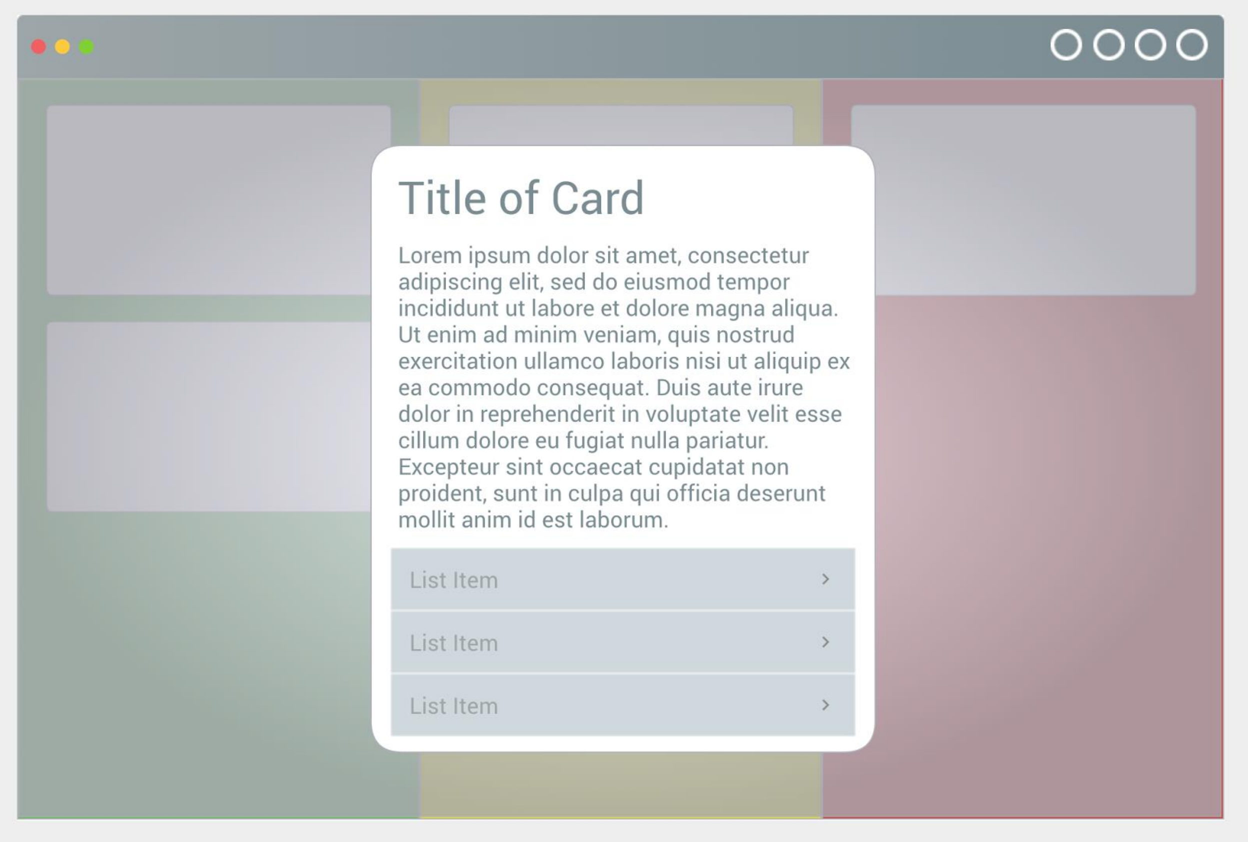
The following section provides an outline of our current front-end platform. From a rudimentary development stage, we have developed a basic UI that will allow a user to view and choose cards on the Toyota themed screen. The following two images demonstrate our basic UI with descriptions regarding the functions and capabilities of each screen.



Title of Card

A short description of the card goes right here. Very simple text that is short sweet and too the point.

Figure 3. This view shows the main table view with its three distinct columns. These columns are used to track the progress of the individual cards/projects. The buttons in the top right will represent board settings, user settings, and other necessary menu items. On the card a short description and a title will be the only things featured.



Title of Card

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Figure 4. This view shows the view when you select a card. You will see more detailed in information for the given project selected. The client/user will be able to add text, lists, and images to the cards to help describe the project and its current state.

Design Pattern

The card/board interaction uses the Front Controller design pattern. User input is handled by the SQL Controller and data output is handled by the Dispatcher. The card and board helpers take the information retrieved by the SQL dispatcher and format it to a structure that the view can easily understand. The view then takes this information and displays it back to the user. An easier way to understand this aspect of our project is to realize that the front end display of cards and boards is a constant and repetitive pattern that will be used again and again in our database via a clever back-end optimization solution.

Testing

This section contains a chart that embodies all our current functional test cases. These cases represent a subset of the final number of test cases that will accumulate during development. However, in our current stage of development, the following test cases are sufficient.

It should be noted that our customer at Toyota has the right to add or remove test cases based on project progress and viability.

Case Number	Test Case	Success Condition	Failure Condition
1.1.1	**name on card exists on click**	card properly displays name when chosen via board	error displays saying a name to card has not been established
1.1.2	**move card on main board**	card moves positions when clicked and dragged	returns error saying card cannot be moved
1.1.3	**exit button on card screen**	exit button returns to main board	exit button returns error saying cannot return
2.1.1	**add text**	add text button opens screen input screen	add text returns erro saying text cannot be added to card
2.1.2	**edit text**	edit text button opens text editor	edit text returns error saying text cannot be edited
2.1.3	**delete text**	delete text button deletes text from card	delete text button returns error saying text cannot be deleted
2.1.4	**edit name**	edit name button opens editor for name	edit name button returns error saying that name cannot be edited

3.1.1	**name storage**	name is properly stored when editor is exited	error is returned saying name could not be saved
3.1.2	**text storage on card object**	text is properly stored when editor is exited	error is returned saying text could not be saved
3.2.1	**card position storage**	position of moved card is stored properly	error is returned saying cards new position could not be saved
4.1.1	**display name on card**	cards name is displayed after change or initiation	error displays in place of blank name
4.1.2	**display text on card**	card text is displayed after change or initiation	error displays in place of blank text
4.2.1	**display cards on board**	card is displayed on board when initiated.	system returns error saying card cannot be created
5.1.1	**store initial account data**	new data input into system is properly stored	error returns saying that new user ID could not be created
5.1.2	**alter initial account data**	editor opens and corrrecly alters ID data	error returns saying user ID data could not be changed
5.2.1	**permissions able to be granted to cards/board**	permissions can be granted to IDs	error returns saying user cannot be granted privileges

5.2.2	**permissions able to be removed to cards/board**	permissions can be removed from IDs	error returns saying user cannot be removed of privileges
5.3.1	**admin established to control/alter permissions**	admin can be established that has all access	error returns that admin cannot be created
5.3.2	**admin can be changed**	admin can be changed following defined protocol	error returns saying correct admin change protocol was not followed
6.1.1	**allows login**	login attempt is successful.	login error message saying ID not found
6.1.2	**success login leads to board**	login displays boards	error returns saying no boards found
6.1.3	**allows new account setup**	new account successfully established to server	error returned saying new account cannot be established
6.2.1	**add name**	name can be added on new user page	error returned that name is taken
6.2.2	**add group**	name can be properly attributed to a group	error returned saying user cannot exist in a group
6.2.3	**leave note for access request**	note is left to admin for access request	error returned saying request could not be sent to admin

7.1.1	**create a list on card**	list feature can be created on a card	error returned saying list cannot be created
7.1.2	**delete a list on card**	list can be deleted from a card	error returned saying list cannot be removed from a card
7.1.3	**edit a list on card**	list can be edited with editor on a card	error returned saying editor cannot be used on list
7.2.1	**store list as data structure on card**	list can be properly saved to a card	error returned saying list could not be saved to a card
7.2.2	**edit data structure if requested**	edit a card if requested by user is successful	error returned saying structure could not be edited
7.2.3	**delete data structure if requested**	delete a card if requested by user is successful	error returned saying structure could not be removed
8.1.1	**create a paragraph on card**	successful creation of a paragraph on card	error returned saying paragraph could not be created on card
8.1.2	**delete a paragraph on card**	successful deletion of a paragraph on card	error returned saying paragraph could not be deleted on card
8.1.3	**edit a paragraph on card**	successful edit of a paragraph on card via editor	error returned saying paragraph could not be edited on card

8.2.1	**store paragraph as data structure on card**	paragraph can be stored onto card	error returns saying paragraph cannot be saved to card
8.2.2	**edit data structure if requested**	structure can be accessed from server	data cannot be accessed and error is returned
8.2.3	**delete data structure if requested**	data structure can be deleted upon request from server	data structure cannot be accessed and error is returned

Figure 5. This figure contains all current test cases. A description of the current scope of test cases is at the beginning of this section.

Review

The following items were reviewed during a team meeting regarding our first sprint and progress made up to this point:

1. Does the basic front-end UI meet general use case?
    - Comment 1: check with Toyota for branding/aesthetic appeal
    - Comment 2: we feel that basic UI is functional and meets spring1 goal
2. Does the back-end store data and is it capable of scalability?
    - Comment 1: We currently see no problems with scalability.
    - Comment 2: check with Toyota for maximum scale of users
    - Comment 3: backend design meets requirements for week 1 sprint goal
3. Is the back-end capable of linking with the front-end UI design flow.
    - Comment 1: As far as we know SQL should seamlessly connect with Electron display.
4. Are there any compatibility flags for Toyota's hardware integration for current program?
    - Comment 1: Ask Toyota for review to detect potential problems during this week's sprint review.
5. Have all components of the architecture assignment been completed to the best of our current project's capacity?
    - Comment 1: It should be noted that our use cases are subject to change, but this is due to unpredictable disruptions and changes during the development cycles.
    - Comment 2: All components have been completed to the best of our ability.
6. Have all aspects of current project been integrated into first presentation?

- Comment 1: It is agreed that all members of team will participate in all presentations. Seamless transitions will be rehearsed.
- Comment 2: All current aspects of project have been correctly incorporated into the presentation.

III. Metrics

Complexity

Our depth inheritance tree is still difficult to predict. Current methods have a very low depth given the basic UI development that has been made. This complexity will magnify once text, name, lists and other data objects are added on top of basic card features. The complexity of the user tree will also become very error-prone when user profiles are created and a network of access is established which will interwire between these classes.

Product Size

We currently have roughly 40 high level test cases to target and expand on. This scope will increase in complexity as the basic test cases are met. Toyota will add more explicit functions to test for once basic UI/backend is developed.

Product Effort

We currently have all put forth roughly 20 man-hours each outside of class. This includes product planning meetings, documentation, website building, travel to Toyota and all communications. This accumulates to a total product effort of 80 hours thus far.

Project Defects

We do not currently have any notable defects in our program.

IV. Site URL/Developer Notebook

1. Project Website URL:

https://github.com/irsyadhanif/CS499Team2Wiki/wiki

2. Effort Log:

Travel and Meeting at Toyota	3 hours
Planning Meetings	8 hours
Website Building	1 hour
Project Planning	4 hours
Architecture	2 hours
Sprint 1	2 hours

20 hours total

**All members followed this same effort schedule up to this point in time.

**We do not currently have any development notes or necessary assumptions.

**All current design decisions were reported in requirements section of this assignment.

3. Team Meeting Log

Meeting with Toyota – August 31, 2017 – 3 hours – All members present

Team Planning Meeting – September 4, 2017 – 2 hours – All members present

Requirements Meeting #1 – September 14, 2017 – 1 hour – All members present

Requirements Meeting #2 – September 16, 2017 – 1 hour – All members present

Sprint Meeting #1 – September 18, 2017 – 45 minutes – All members present

Architecture Meeting #1 – September 25, 2017 – 45 minutes – All members present

Architecture Meeting #2 – September 29, 2017 – 1.5 hours – All members present