ABB KANBAN BOARD

DESIGN DESCRIPTION

GROUP 1

November 30, 2017.

Table of Contents

1. Short Background

ABB Ports is a company that delivers automation and electrical systems for container and cargo handling. Christoffer Holmstedt is currently the lead of an engineering team at this company. Right now, his team is working with *TeamForge* by Collabnet for project administration. This software allows them to have a kanban board for their daily tasks management. Nevertheless, *Teamforge* does not fulfill all the necessities and functionalities needed. Their current solution is to use post-it notes on a whiteboard, but what they really want is a system that allows them to easily create custom categories, swimlanes and that have a simple interface for using it in daily meetings.

Since the system is going to be used every morning on a touch screen display, the client asked for a web application. This allows the ABB team to see the Kanban board in any computer desired. Therefore, the main structure of the web app is going to be developed in HTML, CSS and JavaScript. The backend will be done using Laravel, which is a PHP framework, with the help of *Teamforge* API.

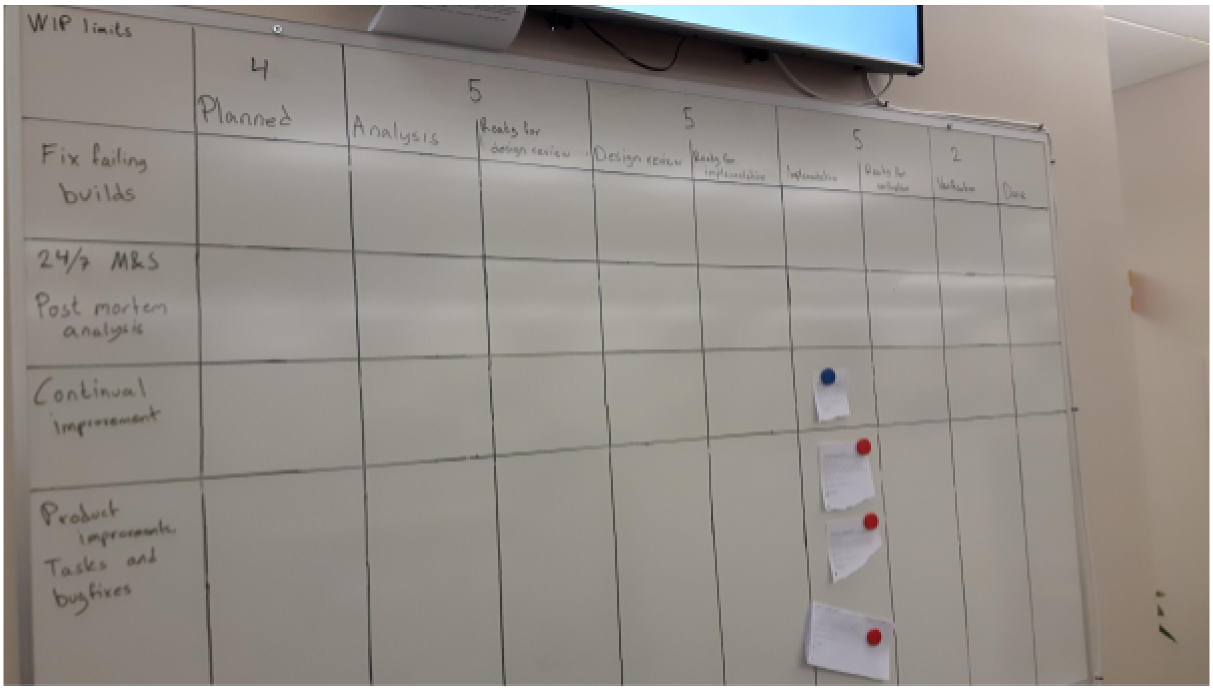


Image 1.1 This is the Whiteboard that Chistoffer's Team is using right now. The design of the system should be similar but also provide an easier way to organize the tasks of the team project.

2. High-Level Description of the System

Our application should represent a *Kanban board* with visualized artifacts imported from TeamForge via its REST API. Each artifact has information that should be presented in our board such as title, description, date, project etc.

Different types of users with different set of activities can use this application such are an ordinary user (team member) and administrator (team leader). Ordinary user is able to *login* and *logout*, to *see the current state* of the Kanban board, and to *drag and drop artifacts* from one category/swimlane to another. Furthermore, administrator (team leader) has additional activities. They are able to *create* both *categories and swimlanes*. Within this activity they can *set card limit* on selected column. Also, they can *filter artifacts* imported from TeamForge.

3. System Overview

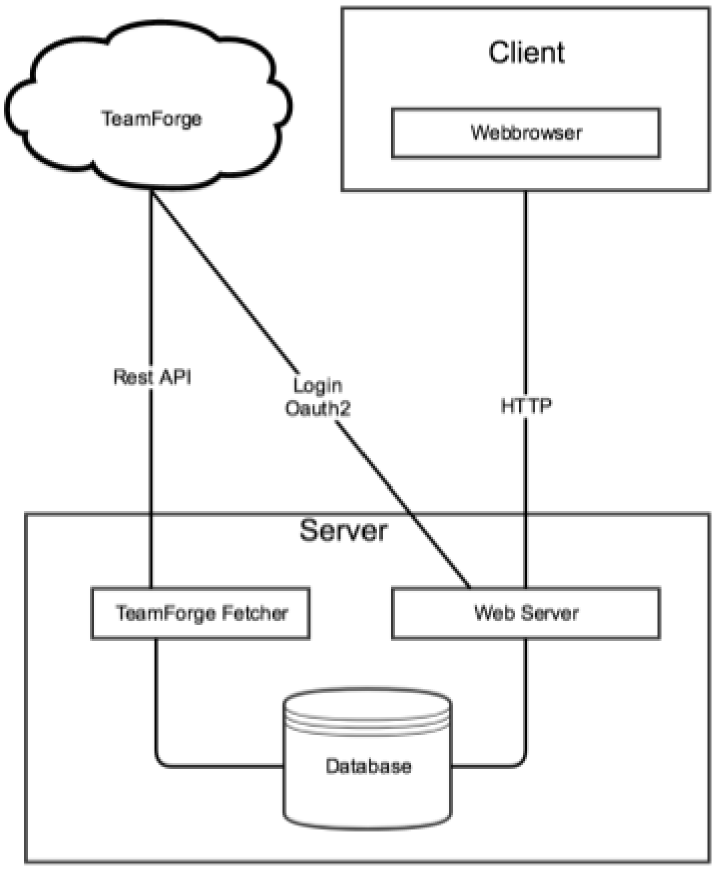


Figure 1. System Architecture Diagram

Our system will be web based and run on a web server. We will use a framework called Laravel. For storing persistent data we will use a MySQL server. Our system will be accessible in the web browsers of the team members at ABB, both on their own computers and a large screen TV.

The system will be based on the *model view controller architecture* described in more details later. We will interface with TeamForge both for getting artifacts from TeamForge and authenticating our clients. For authenticating the users of our system TeamForge provides OAuth2.[1] For fetching artifacts TeamForge provides a REST API.

Artifacts from TeamForge will be fetched and stored in a MySQL database. The artifacts will be updated continuously to reflect the changes made in TeamForge. This will be contained in a separate module, which could be written in a different language and run on a cron job. However, it will most likely run on the web server using the Laravel framework for consistency. If deemed, necessary the module will provide the ability to force an update. Figure 1. presents System Architecture Diagram.

4. Software Architecture

4.1 Decomposition

The major components of our system are *the login system, the database, the view and filtering of TeamForge, the import from TeamForge* and *the Kanban board itself*. The login system passes credentials on to the TeamForge server in an attempt to get a login token from there, which will then be used throughout our system as well. If TeamForge is down or for any other reason doesn't create the token, the users can authenticate with local credentials, different to teamforge, to get access to the board. If that were to happen, the only thing that the user can do is view the kanban board and move the cards around. No new artifacts can be imported.

The admin select what projects should be imported from the TeamForge REST API and it updates our database with new and modified artifacts. An admin can then find and decide what artifacts should be displayed on the board using a filter module. The database is regularly updated so that the stored items reflect the information on the TeamForge server. The kanban board reads its state from the database and updates the database whenever a user makes a change to the board.

[1] http://help.collab.net/index.jsp?topic=/teamforge174/faq/teamforge_oauth_2-0.html

4.2 Persistent data

Our database stores data such as attributes of categories and swimlanes, attributes of local users, and imported attributes of artifacts and available projects from the TeamForge API. Furthermore, cards representing stored information about artifacts are also stored in the database. A card can only represent a single artifact. Entity relationship diagram represents these models with its data in detail (Figure 3.).

An administrator is able to manipulate this data by creating, updating and deleting categories and swimlanes. Administrator imports and filters artifacts from TeamForge as well. Since there is no sync back with TeamForge, admin is not able to create, update or delete artifacts from the database using our web application. Other users do not have permission to change these attributes within the database.

4.3 Synchronization and timing

Existing concerns for synchronizations and timing are:

- Drag and drop functionality, when several users trying to move the same card/cards at the same time but to different columns. If that were to happen the cards will be send as a queue and the board will update continuously The second person who hold the same card and move to another column then the position of the card will be updated again.

- After moving a card to another column the board should automatically update without refreshing the entire web page. The board will be checking for updates continuously every three seconds and if there are any updates then it will refresh.

- Limited login session can be annoying, the user have to refresh the page or login again for accessing the page.

- Updated/removed information in TeamForge will affect the Admin dashboard and the Kanban board. A concern is how to handle this. A solution is that all of the information that are removed from TeamForge will be removed from our database.

- Losing connection to TeamForge, the admin dashboard can’t be updated.

4.4 Security

Security is important for our client since the data they work with is confidential. Most of the security concerns with our software are taken care of by Laravel. Our software’s user accounts are managed by TeamForge using OAuth2 which is considered secure.

Most of the security concerns are with the environment which our software runs on. When deploying our software the client can take security measures according to their own needs. Some of the security measures could be: configure database users and limit database access, put the webserver on it’s own isolated network or only allow local access.

Since we only fetch data from TeamForge, our software should not do any harm, other than disclosing secret data. However, if our software gets compromised an intruder could get access to the TeamForge OAuth2 token we use to communicate with TeamForge’s REST API. Access to the token would grant the malicious user the ability to modify and delete data from TeamForge.

5. Detailed Software Design

5.1 Model-view-controller

Model-view-controller (MVC) is a design pattern we are using in our web application (Figure 1.). It was originally developed by *Smalltalk* programmers. The design pattern divides the application into three interconnected parts. This is done to separate the external representation of information from the way it's internally stored. These three parts are as the name suggests the *model*, the *view* and the *controller*.

A view is a representation of information to the user. For a web application this is the part that generates the web page the user sees. A model's job is to represent the underlying data, rules and logic of the application while being independent from the user interface. Finally, a controller represents the connection between the model and the view. It accepts input and converts it into commands which it sends to either of them to make them do something. These three parts interact to allow the user to use the application. It also gives the developer more options for efficient code reuse and parallel development.

Laravel's design indirectly makes us use the MVC design pattern while developing. While it isn't fully a MVC framework it does take inspiration from the design pattern. It has the concepts of models, views and controllers that mimic their respective counterparts in MVC.

Our project has models for *Artifact, Card, Category, Parent Category, Project, Swimlane, User*. These will communicate directly with the database to allow our views to get and manipulate their respective data.

We currently have the following views:

- *Login* view;
- *Kanban board* view enables both users and administrators to preview the current state of the board, and, drag and drop cards to the new state;
- *Swimlane* view where the administrator is able to create/edit/remove a swimlane;
- *Category* view where the administrator is able to create/edit/remove a category;
- *Parent category* view where the administrator is able to create/edit/remove a parent category;
- *Projects* view where the administrator is able to import projects from TeamForge REST API and select from which projects should artifacts be imported;

- *Filter artifacts* view where the administrator is able to update available artifacts with TeamForge REST API and select which artifacts should be shown in the *Kanban Board* view;

The user has access to the login view and kanban board view. In addition to these, the admin has access to views for overlooking, creating and editing the categories, swimlanes, parent categories, and the views for importing projects and filtering artifacts.

These views are accessible via their respective routes. The route routes the user's request to its respective controller which then renders the requested view.

5.2 Class Diagram

Figure 2. presents Class Diagram.

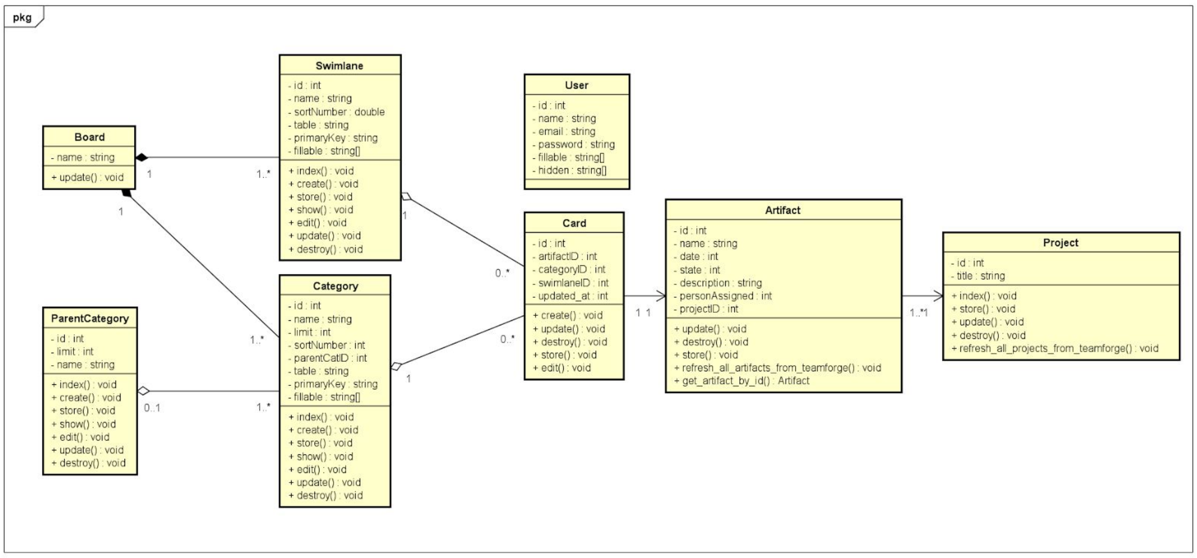


Figure 2.

Not every class in the entire application is in the class diagram. This is because of our use of the framework Laravel. A class diagram containing every single class that is used would simply be so large as to become incomprehensible. Some information about how the framework operates can be found in other parts of this document, or in the official Laravel documentation. The classes that are in this class diagram are a compilation of the classes that we have written. Some classes may have been merged together to increase the ease of understanding.

In this class diagram we have eight classes. Starting from the top left we have the Board class. This class will represent the Kanban board. The ParentCategory class represents a collection of Categories. Not every Category need to have a ParentCategory, it is optional. The Category class represents the vertical division, while the Swimlane class represents the horizontal. A Kanban board needs to have at least one Category, but may have no Swimlanes.

On the right side there is the Project class represents a project. The User class will represent the users. The Artifact class will hold all of the information gathered from the TeamForge API, but will not show on the actual board. What will actually show on the board is instances of the Card class. The Card class will hold the position of the card, and a foreign key to the Artifact that it represents.

5.3 Entity Relationship Diagram

Figure 3. presents our Entity Relationship Diagram (ERD).

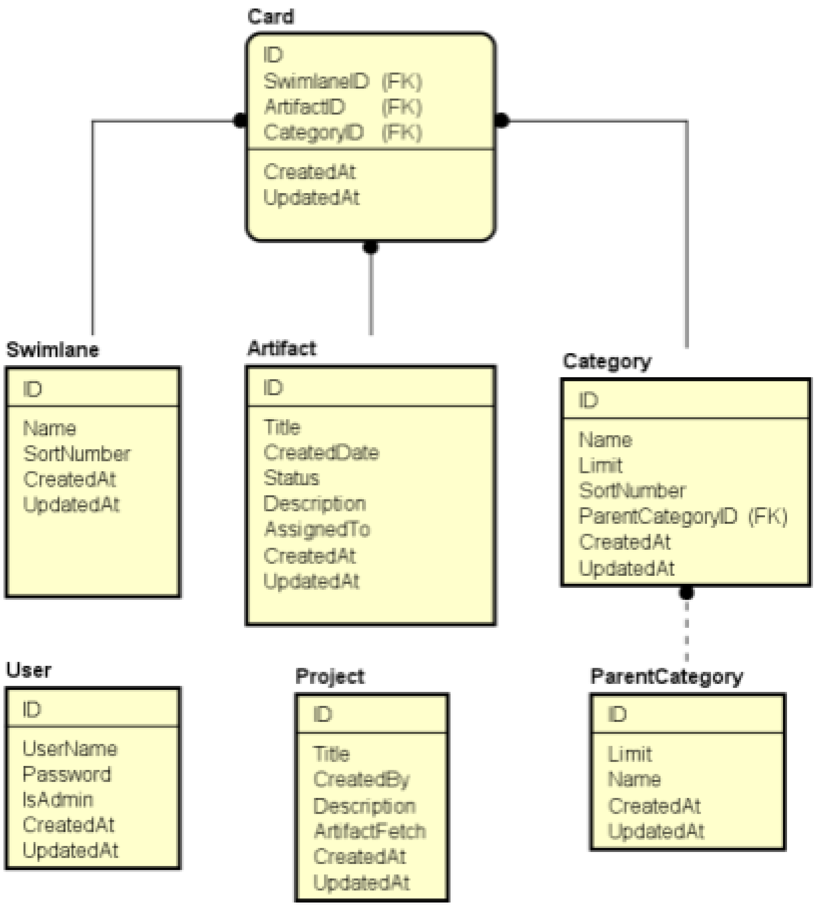


Figure 3. Entity Relationship Diagram

Due to security reasons, all artifacts will not be available for import in web application. Administrator will have to select a set of projects of which all artifacts can then be imported to our application. From that starting point, *Project* is the primary table for establishing a good structure for our application. Each project will have its title and as many artifacts as possible. *Artifact* model table has many to one relation with *Project* meaning that one artifact can be part of only one project while one project can have many artifacts.

Next step in modelling the artifact and presenting it on Kanban board is creating the model/table of *Card* which holds the information on where the certain artifact is on the Kanban board. One artifact can belong to many instances of card but one card has only one artifact it holds. Card then holds information on *Swimlane* which is shown in the ERD as well as *Category*. Category is connected to the *ParentCategory* in order to model the opportunity to have categories within categories. Table *User* is used to store user information of local users used in case TeamForge is down.

6. Graphical User Interface

The Kanban board web application will consist of several web pages with different options on available on them.

These are:

- Login page
- Kanban board
- Administrator dashboard
    - Import and filter artifacts page
    - View/edit/remove/add swimlanes page
    - View/edit/remove/add categories page
    - View/edit/remove/add parent categories page
    - View/select/unselect/update projects page

The landing page for our application will be a login page. This is due to the fact that this application is for in-house (ABB Ports) use only meaning that all of the functionalities are available only to ABB ports employees. The login page will offer user authentication via TeamForge login service as well as using username and password from a local application database if TeamForge is not available.

After the user is logged in, he or she is presented with a Kanban board, which is the main functionality of this web application. If the user logged in is an ordinary user he or she will be able to view current state of Kanban board, see more information about a card, as well as move the cards over the board. The Kanban board will be spread over the whole page (web browser) with a navbar at the top where the user will have an option to logout when finished using Kanban board.

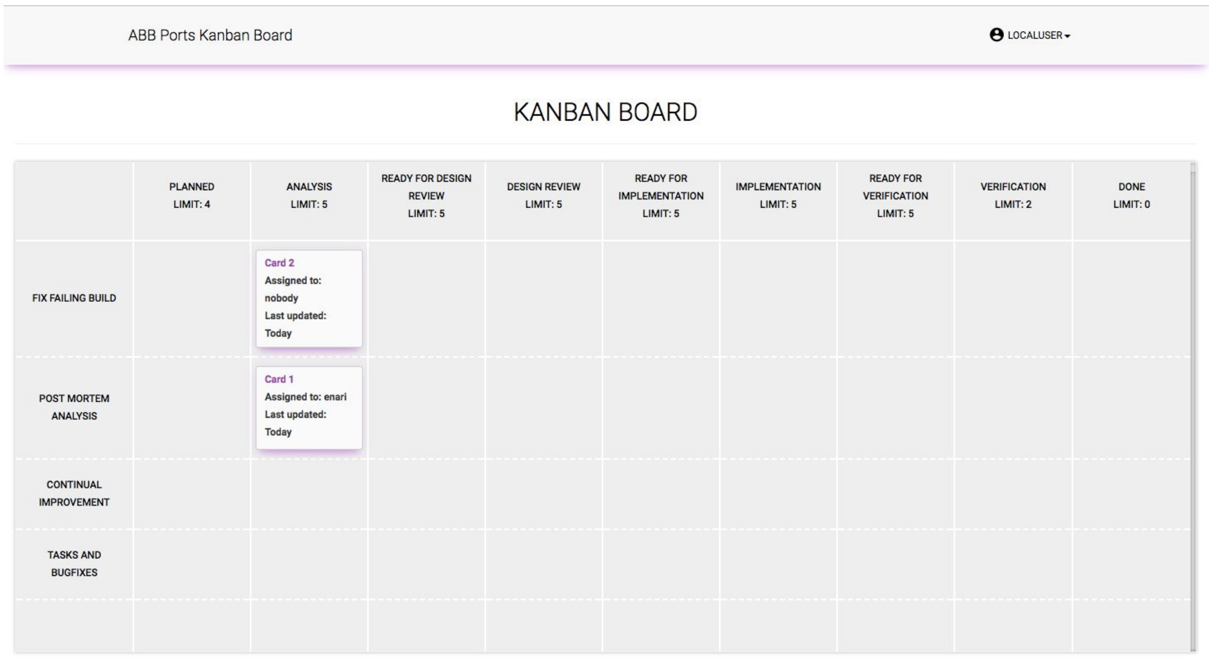


ABB Ports Kanban Board

LOCALUSER

KANBAN BOARD

	PLANNED LIMIT: 4	ANALYSIS LIMIT: 5	READY FOR DESIGN REVIEW LIMIT: 5	DESIGN REVIEW LIMIT: 5	READY FOR IMPLEMENTATION LIMIT: 5	IMPLEMENTATION LIMIT: 5	READY FOR VERIFICATION LIMIT: 5	VERIFICATION LIMIT: 2	DONE LIMIT: 0
FIX FAILING BUILD		Card 2 Assigned to: nobody Last updated: Today							
POST MORTEM ANALYSIS		Card 1 Assigned to: enari Last updated: Today							
CONTINUAL IMPROVEMENT									
TASKS AND BUGFIXES									

Figure 4: View for an ordinary user/local user

If the logged in user is an administrator, then the options of web application are extended. Administrator will have the same functionalities available for him or her over Kanban board as for the ordinary user, with an additional option of visiting an admin dashboard.

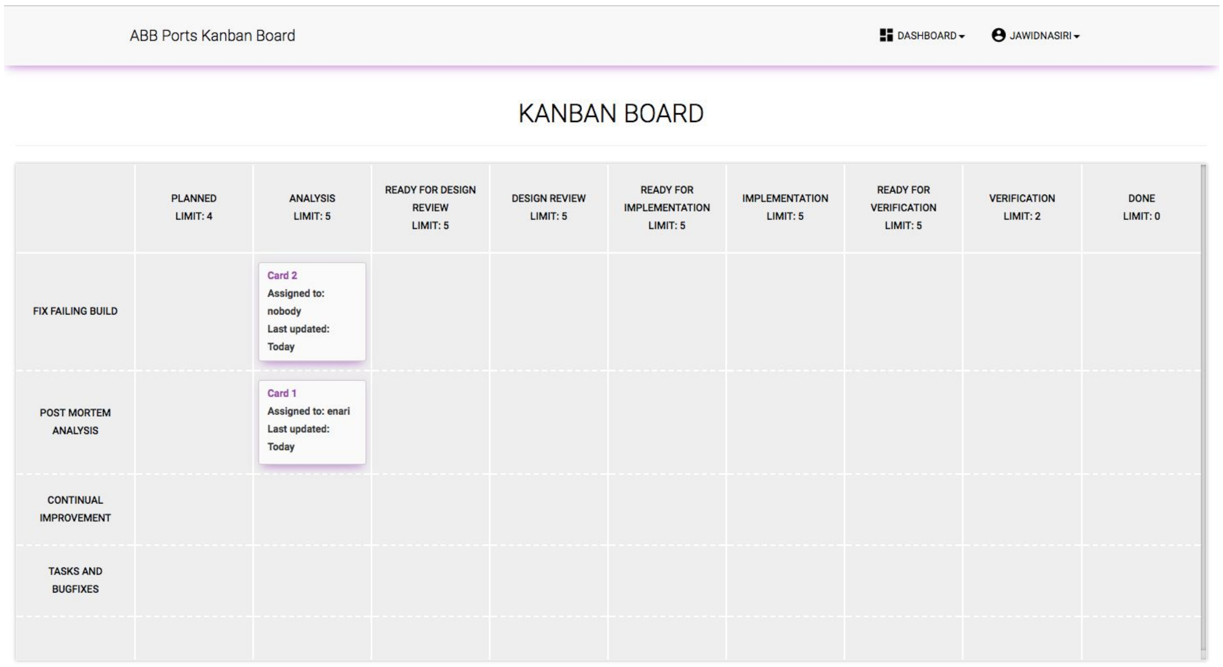


ABB Ports Kanban Board

DASHBOARD JAWIDNASIRI

KANBAN BOARD

	PLANNED LIMIT: 4	ANALYSIS LIMIT: 5	READY FOR DESIGN REVIEW LIMIT: 5	DESIGN REVIEW LIMIT: 5	READY FOR IMPLEMENTATION LIMIT: 5	IMPLEMENTATION LIMIT: 5	READY FOR VERIFICATION LIMIT: 5	VERIFICATION LIMIT: 2	DONE LIMIT: 0
FIX FAILING BUILD		Card 2 Assigned to: nobody Last updated: Today							
POST MORTEM ANALYSIS		Card 1 Assigned to: enari Last updated: Today							
CONTINUAL IMPROVEMENT									
TASKS AND BUGFIXES									

Figure 5: View for an administrator

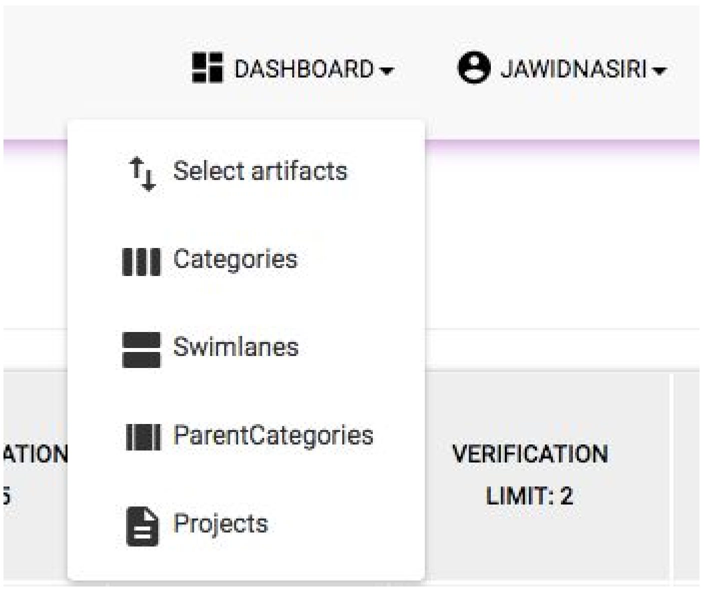
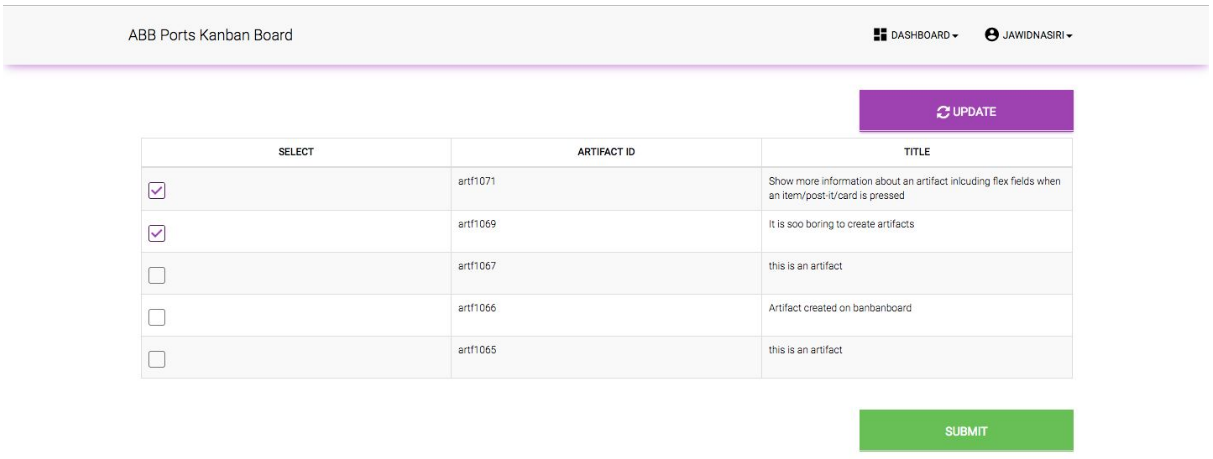


Figure 6: Administrator dashboard

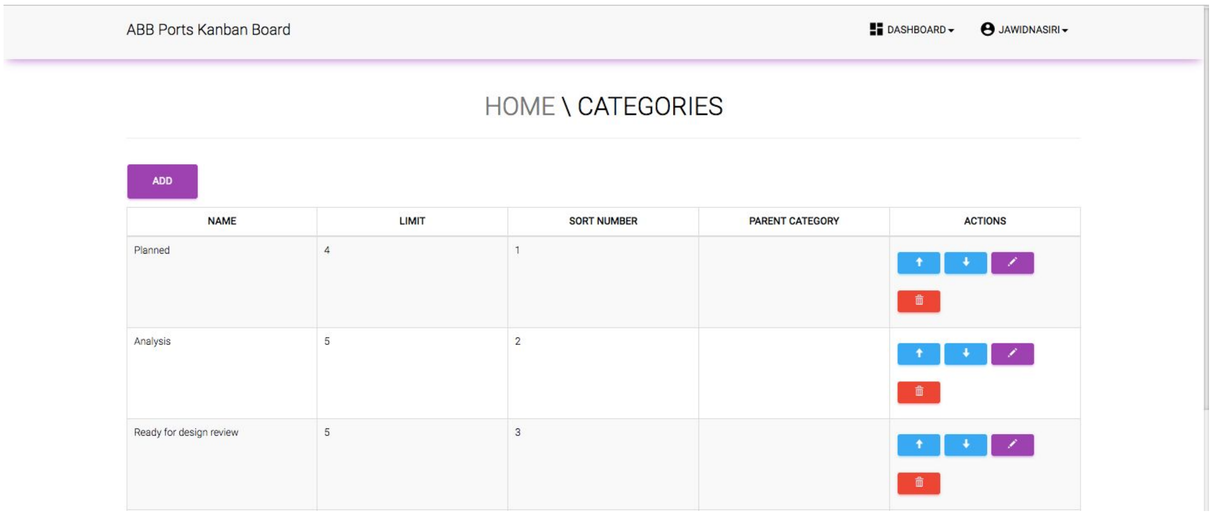
In the admin dashboard there will be links available for five different pages. First page, the landing page for admin dashboard is import and filtering of artifacts from TeamForge. Here, administrator will have an option to import artifacts based on the project and to select which of them appear on the board.



SELECT	ARTIFACT ID	TITLE
✓	artf1071	Show more information about an artifact inlcuding flex fields when an item/post-it/card is pressed
✓	artf1069	It is soo boring to create artifacts
☐	artf1067	this is an artifact
☐	artf1066	Artifact created on banbanboard
☐	artf1065	this is an artifact

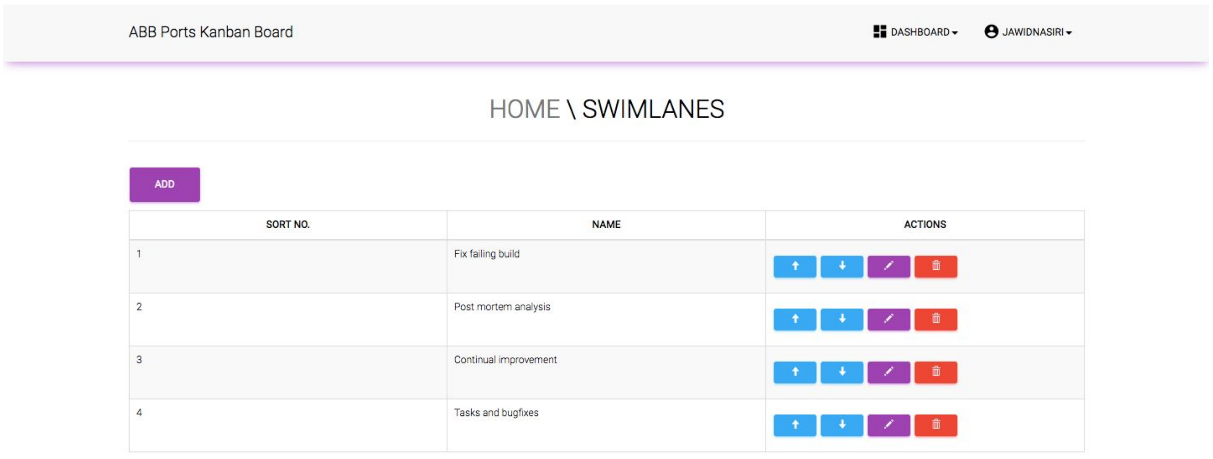
Figure 7: Select artifact page

Second and third page available from admin dashboard are pages where admin will be able to add, remove and edit swimlanes and categories of the Kanban board.



NAME	LIMIT	SORT NUMBER	PARENT CATEGORY	ACTIONS
Planned	4	1		
Analysis	5	2		
Ready for design review	5	3		

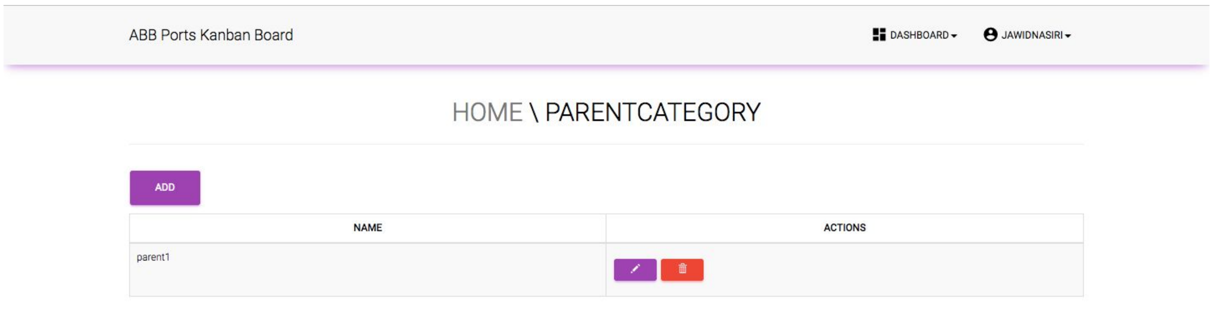
Figure 8: Category page



SORT NO.	NAME	ACTIONS
1	Fix failing build	
2	Post mortem analysis	
3	Continual improvement	
4	Tasks and bugfixes	

Figure 9: Swimlane page

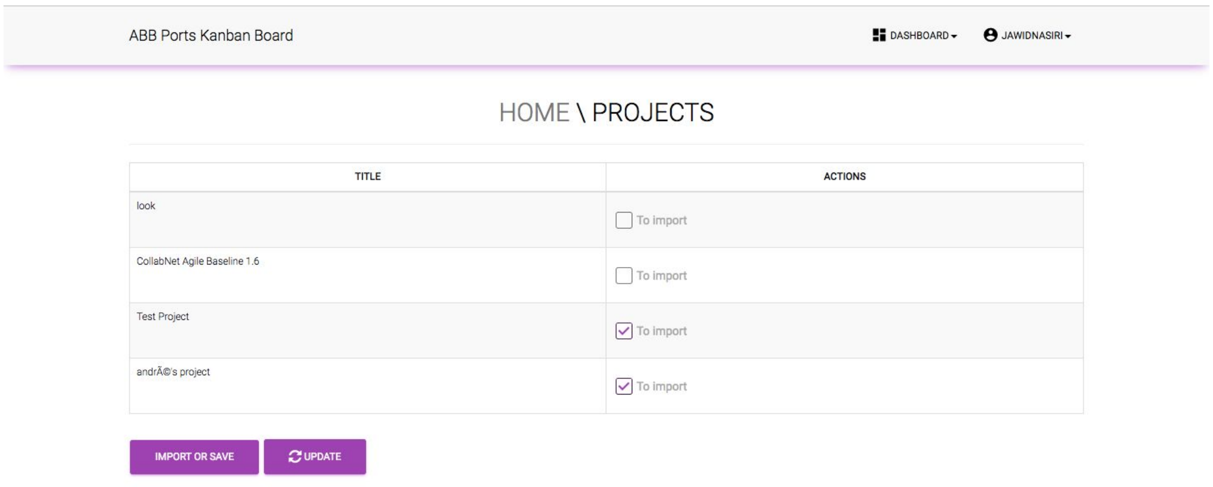
The fourth page is for add new parent category which allow the administrator to add a new parent category or modify the existed.



NAME	ACTIONS
parent1	

Figure 10: Parent category page

The last page is projects, in this page the administrator will see a list of all projects imported from TeamForge. The project page gives the ability to select and unselect a project or update. After selecting a project and submit the page will redirect the administrator to select artifact page that allows to pick artifact for displaying on the board.



TITLE	ACTIONS
look	☐ To import
CollabNet Agile Baseline 1.6	☐ To import
Test Project	✓ To import
andrÃ©'s project	✓ To import

Figure 11: Project page