Bovidae Framework

By:

Kainat Khalid
15178
Asma Malik
20939
Amna Shehzadee
15395

Supervised by:
Ms. Sumera Saleem
(Assistant Professor)

Faculty of Computing
Riphah International University, Islamabad

Spring 2015

A Dissertation Submitted To

Faculty of Computing,

Riphah International University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of

Bachelors of Science in Software Engineering

Faculty of Computing
Riphah International University, Islamabad

Final Approval

This is to certify that we have read the report submitted by ***Kainat Khalid (15178)***, ***Asma Malik (20939)***, ***Amna Shehzadee (15395)*** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Software Engineering (BSSE). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Software Engineering (BSSE).

Committee:

1 ________________________________

Ms. Sumera Saleem(Assistant
Professor)
(Supervisor)

2 ________________________________

Ms. Sumera Saleem
(Head of Department/chairman)

Declaration

We hereby declare that this document "**Bovidae Framework**" neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanying report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **Ms. Sumera Saleem (Assistant Professor)**. If any part of the system is proved to be copied out from any source or found to be a reproduction of any project from anywhere else, we shall stand by the consequences.

Kainat Khalid

15178

Asma Malik

20939

Amna Shehzadee

15395

Dedication

To the kindest person, Mother and the dearest person, Father;
Who were always our support and whose prayers brought us up to this level,
And
To our supervisor;
The kindest and dedicated person without his motivation we cannot accomplish this project.

Acknowledgement

First of all we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

We are wholeheartedly thankful to **Ms. Sumera Saleem (Assistant Professor),** our respected and most dedicated supervisor whose guidelines are helping in accomplishing our project.

Kainat Khalid

15178

Asma Malik

20939

Amna Shehzadee

15395

Abstract

PHP has been the language of the web for a very long time due to its ease of learning, community, is free and open source and ease of deployment. Over the last few years, the web development industry has seen groundbreaking changes in Frameworks. PHP framework tries to ease the development process and reduce the number of errors. As the time has gone popular PHP frameworks like Laravel, Phalcon, Symfony and CodeIgniter have just got bigger and better and become a handy tool for developers to build giant application effortlessly. So we propose framework architecture "Bovidae Framework". Bovidae Framework will be a powerful PHP framework which is built for developers who need a simple and elegant toolkit to create full-featured web applications. Bovidae Framework is Simple solutions over complexity. Bovidae Framework is an easy way to build full-featured web applications.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Chapter 1:

Introduction

Bovidae Framework will be a powerful PHP framework which is built for developers who need an easy way to create full-featured web applications. Bovidae Framework is Simple solutions over complexity. Bovidae framework also adds structure to the code, prompting the developer to write better, more readable, and more maintainable code. Ultimately, this framework makes programming easier, since it packages complex operations into simple statements.

For a simple web application or a website with basic features, it's wiser to use some good open source system. However, for more complex and bespoke functionality requirement it's more recommended to work with the framework. A completely customized structure, functionality developed keeping in mind your business and users, can be much handier rather than enhancing off the shelf systems which can many times be unnecessarily huge, confusing and difficult to manage. The key to a successful bespoke system built on any PHP framework is, to well plan the architecture and approach following the best development practice resulting in something more unique to business requirements.

The huge amount of development time can be saved with the use of Frameworks due to the availability of the large variety of tools like input sanitization and abstraction layers in particular. It's a very common scenario in website development companies where clients are really pressurizing and so often demand quick turnaround. Frameworks can be real lifesavers under high pressure as they help enormously in quickly delivering repetitive and commons tasks.

There are pretty good reasons to use the Frameworks:

- Code and file organization is extremely easy
- Countless numbers of tools and libraries that can help you with:
    - Form validation
    - Database abstraction
    - Input/output filtering

    - Session and Cookie handling
    - Email, Calendar and pagination and much more
- MVC (Model View Controller) Architecture
- Less code which ultimately speeds up your development
- Security - PHP has many input and output filtering functions which can add an extra security layer to protect your website against certain attacks
- Suitable for teamwork
- PHP frameworks have great active community support. You will find the accurate solution of any query quickly

1.1 Background:

A PHP Framework allows us to develop web applications. PHP framework tries to ease the development process and reduce the number of errors but Existing system depends on 3rd party vendors, it’s very difficult and time taking to understand their terms and techniques. Some frameworks take a lot of time, at the time of deployment that results in wastage of time of a user. Some framework has an absence of continuation between the versions because of this reason user face a lot of trouble in converting from one version to other. Dependency injection is complex because of its complexion it makes the code more difficult and more coupled. When documentation is accurate and complete, it works wonders but when documentation is not accurate it cause many problems like some framework have documentation lacks and because of this reason new developer takes a lot of time to understand it. It is difficult for them to maintain or modify codes. Some frameworks take more memory and also quite slow.

It is highly frustrating trying to learn a new technology that hasn’t been fully documented. Over recent years, this has been a big flaw with a lot of the new PHP frameworks. It seems to be the norm that framework developers expect early adopters to reverse engineer classes to fully understand their capabilities. This adds a giant learning curve and creates an initial lag in development as time is wasted on figuring out what is, and isn't, possible. Documentation is clearly very important for developers. From the get go, one of Bovidae goals was to offer up-to-date and useful documentation, paving the way for a much smoother transition to adopting the framework.

1.2 Motivation and Challenges:

Our System is somehow different from other systems as are our Bovidae Framework will be a powerful PHP framework which is built for developers who need an easy way to create full-featured web applications. Bovidae Framework is Simple solutions over complexity. The most important factor i.e., the huge amount of development time can be saved with the use of Frameworks due to the availability of the large variety of tools and it offers up-to-date and useful documentation, paving the way for a much smoother transition to adopting the framework.

1.3 Goals and Objective:

The basic aims and objectives of Bovidae – An easy way to build full-featured web applications:

- This framework makes the development process quite easier. It takes very little time to complete the web project with the help of this framework.
- You can also create different routes
- Makes implementing authentication very simple. Almost everything is configured out-of-the-box
- it helps to secure the web application by protecting it against the most serious security risks: SQL injection, cross-site request forgery, and cross-site scripting
- Error and exception handling is already configured.
- Separation of Business logic code from presentation code
- It also provides template engine, by which you don't need to use server-side code in your view file
- It simplifies the web development process by easing the common tasks

1.4 Solution overview:

Over the last few years, web development industry has seen ground breaking changes in frameworks. As the time has gone popular PHP frameworks like Laravel, Phalcon, Symfony, Zend, and Code Igniter have just got bigger and better and become a handy tool for developers to build giant application effortlessly. PHP is one of the widely used languages for web development and almost every reputed CMS system using PHP as a base programming language. In PHP a huge amount of development time can be saved with the use of Frameworks. Today the most famous frameworks like Laravel, Phalcon, Symfony, Zend, and CodeIgniter still have some drawbacks. Developer faces some challenges while developing websites. We purpose a framework architecture, this framework has an ability to resolve all the problems of PHP framework. It has the following features:

- Well Organized File Architecture
- Form Validation
- Database Abstraction
- Input / Output Filtering
- Session and Cookie Handling
- Integration of Different Email Clients (like PHP Mailer etc.)
- Separate Server Side language from Client Side by using Template Engine
- Auto Model Class Generations
- Less Code which speeds up your development
- Reduce the occurrence of an error
- Security
- It also has ORM which allows us to represent or map relational database and generate SQL statements
- Simpler documentation
- It uses a blade template engine to speed up compiling tasks, and users can include the latest features so easily.

1.5 Report outline

This report covers in details all the aspects of the system. For the sake of understanding and clarity, this report has been divided into seven different chapters:

Chapter 1: Introduction

Chapter 2: Literature Survey

Chapter 3: Requirement Analysis

Chapter 4: System Design

Chapter 5: Implementation Chapter

Chapter 2
Literature/Market Survey

2 Literature market survey

2.1 Introduction

In this chapter, existing systems similar to “Bovidae Framework “are discussed, what they are lacking and where they need improvement. All the framework running around the world for web application services are not vast, many of the facilities are missing that can be provided to the developer and failed to provide most facilities under one platform. The crucial issues need a system like of ours - to ease the developer. Frameworks need to figure out more facilities that will overcome the developer’s issue.

2.2 Literature Review/Technologies Overview

There are frameworks services that are web-based. They are working but still have some incomplete services for developers. Although there are many frameworks, providing services for developers but there is no specific system that provides all services under a single platform. No. other system is bothering that one must compile all the developer’s needs under the single platform which could assist the developer to a great extent.

Existing frameworks have some issue although they have some benefits but still are some technical issue exists. Following are discussed below:

➢ **Introduction to Laravel**

Laravel is a web application framework with an expressive, elegant syntax. Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality.

- **Advantages of Laravel framework**
  - ✓ Best for any Project (Small, Medium, Large).
  - ✓ Run on PHP>=5.6 version
  - ✓ Multiple databases
  - ✓ Clear Documentation

    - ✓ Laravel has good speed.
    - ✓
- **Disadvantages of Laravel framework**
    - ✓ High complexity in installment and adjustment
    - ✓ Complex deployment
    - ✓ Large folder size
    - ✓ Dependency Injection Container

➢ **Introduction to Codeigniter**

Codeigniter is an Application Development Framework - a toolkit - for people who build websites using PHP. Its goal is to enable you to develop projects much faster than you could if you were writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries. Codeigniter lets you creatively focus on your project by minimizing the amount of code needed for a given task.

- **Advantages of Codeigniter framework**
    - ✓ Best for any Project (Small, Medium, Large).
    - ✓ Run on PHP5.1 version
    - ✓ Multiple databases
    - ✓ Clear Documentation
    - ✓ Codeigniter2 has average speed
    - ✓ Low complexity in installment and adjustment
    - ✓ Low Dependency Injection Container
- **Disadvantages of Codeigniter framework**
    - ✓ Complex deployment
    - ✓ Average folder size

➢ **Introduction to Symfony**

Symfony2 is a reusable set of standalone, decoupled and cohesive PHP components that solve common web development problems.

- **Advantages of Symfony framework**

- ✓ Best for any Project (Small, Medium, Large).
- ✓ Run on PHP5.3 version
- ✓ Multiple databases
- ✓ Clear Documentation

- **Disadvantages of Symfony framework**
  - ✓ Complex deployment
  - ✓ Best only for large projects.
  - ✓ Dependency Injection Container
  - ✓ High complexity in installment and adjustment
  - ✓ Symfony2 has slow speed
  - ✓ Average folder size

➢ **Introduction to Yii**

Yii2 is a free, open-source framework for PHP5 that promotes clean, DRY design, and supports rapid development.

- **Advantages of Yii framework**
  - ✓ Best for any Project (Small, Medium, Large).
  - ✓ Run on PHP5.4 version
  - ✓ Multiple databases
  - ✓ Clear Documentation
  - ✓ Average folder size
- **Disadvantages of Yii framework**
  - ✓ Complex deployment
  - ✓ Average complexity in installment and adjustment
  - ✓ Yii2 has slow speed
  - ✓ Dependency Injection Container

➢ **Introduction to Phalcon**

Phalcon is an open source full-stack framework for PHP, written as a C-extension. Phalcon is optimized for high performance. Its unique architecture allows the framework to always be memory resident, offering its functionality whenever it's needed.

- **Advantages of Phalcon framework**
  - ✓ Best for any Project (Small, Medium, Large).
  - ✓ Run on PHP5.4 version
  - ✓ Multiple databases
  - ✓ Clear Documentation
  - ✓ Low complexity in installment and adjustment
- **Disadvantages of Phalcon framework**
  - ✓ Complex deployment
  - ✓ Dependency Injection Container
  - ✓ Phalcon has average speed
  - ✓ Average folder size

The biggest problem is that many of the problems may have the solution in any of the above-mentioned systems but none Frameworks provides all facilities under one platform. Our framework bestows an appropriate solution to such problems that all such issues would be resolved under one portal facilitating in maximum ways to the developer.

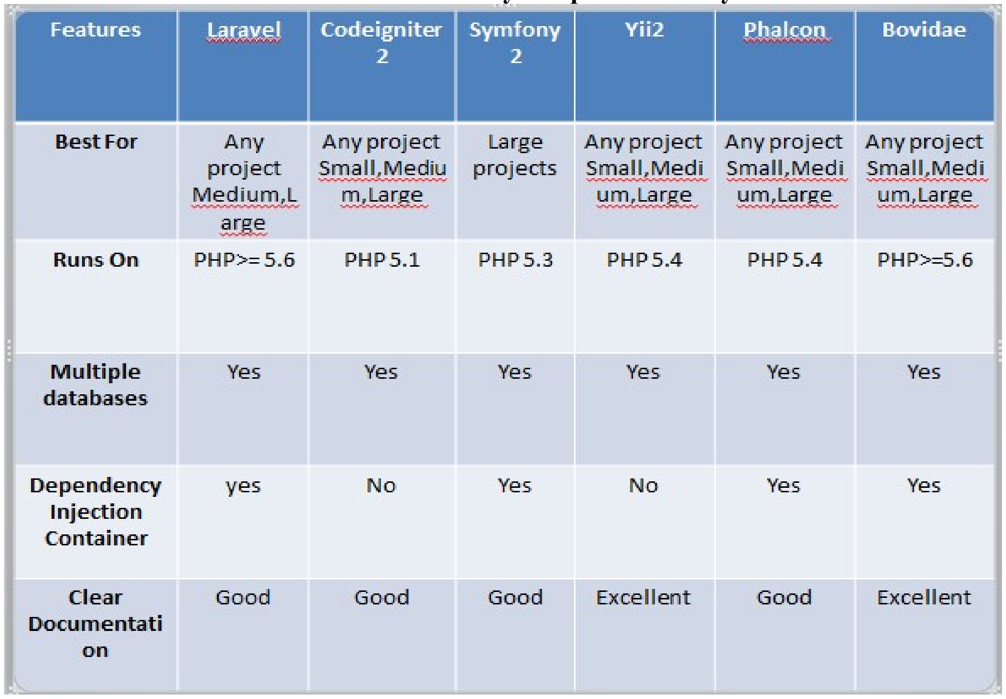
➢ **Introduction to Bovidae**

Bovidae Framework is a complete system which is resolving the most issues of the developer. It is a framework that has less deployment time, faster speed and so on. All in all, the Bovidae Framework is a perfect blend of features that will make lives easier for the developer.

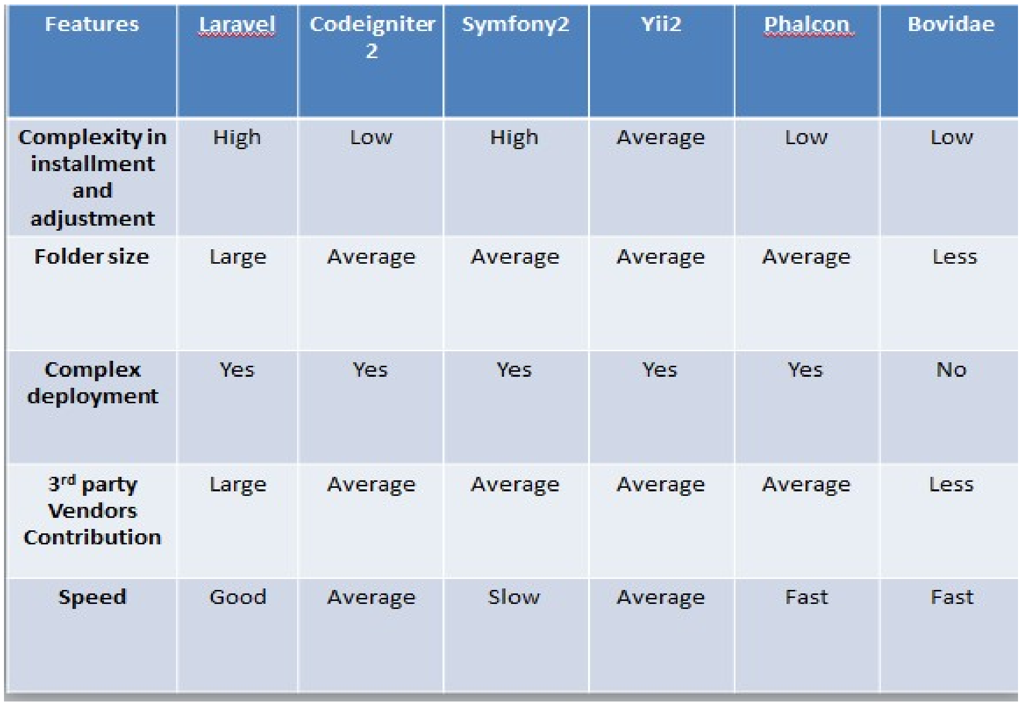
- **Advantages of Bovidae framework**
  - ✓ Best for any Project (Small, Medium, Large).
  - ✓ Run on PHP>=5.6 version
  - ✓ Multiple databases
  - ✓ Clear Documentation
  - ✓ small folder size
  - ✓ Low complexity in installment and adjustment
  - ✓ Bovidae will have fast speed

- ✓ Low Complex deployment
- ✓ Low Dependency Injection Container

Table 1: Market Survey comparative analysis



Features	Laravel	Codeigniter 2	Symfony 2	Yii2	Phalcon	Bovidae
Best For	Any project Medium,Large	Any project Small,Medium,Large	Large projects	Any project Small,Medium,Large	Any project Small,Medium,Large	Any project Small,Medium,Large
Runs On	PHP>= 5.6	PHP 5.1	PHP 5.3	PHP 5.4	PHP 5.4	PHP>=5.6
Multiple databases	Yes	Yes	Yes	Yes	Yes	Yes
Dependency Injection Container	yes	No	Yes	No	Yes	Yes
Clear Documentation	Good	Good	Good	Excellent	Good	Excellent



Features	Laravel	Codeigniter 2	Symfony2	Yii2	Phalcon	Bovidae
Complexity in installment and adjustment	High	Low	High	Average	Low	Low
Folder size	Large	Average	Average	Average	Average	Less
Complex deployment	Yes	Yes	Yes	Yes	Yes	No
3rd party Vendors Contribution	Large	Average	Average	Average	Average	Less
Speed	Good	Average	Slow	Average	Fast	Fast

2.3 Summary

The market survey tells us that very few frameworks is providing technical services. Other Framework fails to provide most of the facilities like speed facility under one platform. The developer has to use different Framework to availing the services. Our Framework ‘Bovidae Framework is an interactive Framework which provides all services to the developer which he/she wants. Now the developer can avail most required services easily.

Chapter 3:
Requirement Analysis

Requirement Analysis

3.1 Introduction:

This chapter specifies the requirements for the "Bovidae Framework". It illustrates the problem statement, functional and nonfunctional requirements, use case diagram and specification of the application and thus covers the scope of the project.

3.2 Problem Scenarios:

PHP has been the language of the web for a very long time due to its ease of learning, community, is free and open source and ease of deployment. Over the last few years, the web development industry has seen groundbreaking changes in frameworks. A PHP Framework allows us to develop web applications. PHP framework tries to ease the development process and reduce the number of errors but Existing system depends on 3rd party vendors, it's very difficult and time taking to understand their terms and techniques. Some frameworks take a lot of time, at the time of deployment that results in wastage of time of the user. Some framework has an absence of continuation between the versions because of this reason user face a lot of trouble in converting from one version to other. Dependency injection is complex because of its complexion it makes the code more difficult and more coupled. When documentation is accurate and complete, it works wonders but when documentation is not accurate it cause many problems like some framework have documentation lacks and because of this reason new developer takes a lot of time to understand it. It is difficult for them to maintain or modify codes. Some frameworks take more memory and also quite slow.

We purpose framework architecture "Bovidae Framework". The bovidae framework has the ability to resolve all the problems discussed before. On Bovidae framework, heavy or small budget web project can be developed.

3.3 Functional Requirements:

A functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. Following are the functional requirements of our system:

See functional requirements in Appendix A

3.4Non- Functional Requirements:

3.4.1 [NF-1] Portability

Bovidae Framework portable for MAC, Windows and for linux Operating systems.

3.4.2 [NF–3] Efficiency

The system shall be Efficient in terms of time behavior.

3.4.3 [NF–4] Support

Our framework generally includes documents, Facebook page, twitter account, Github account and YouTube channel.

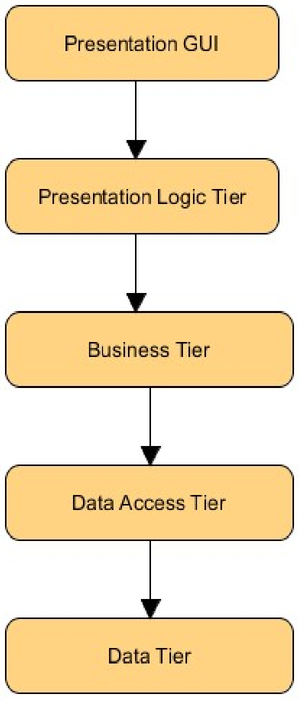
Chapter 4:
System Design

4.1 Introduction:

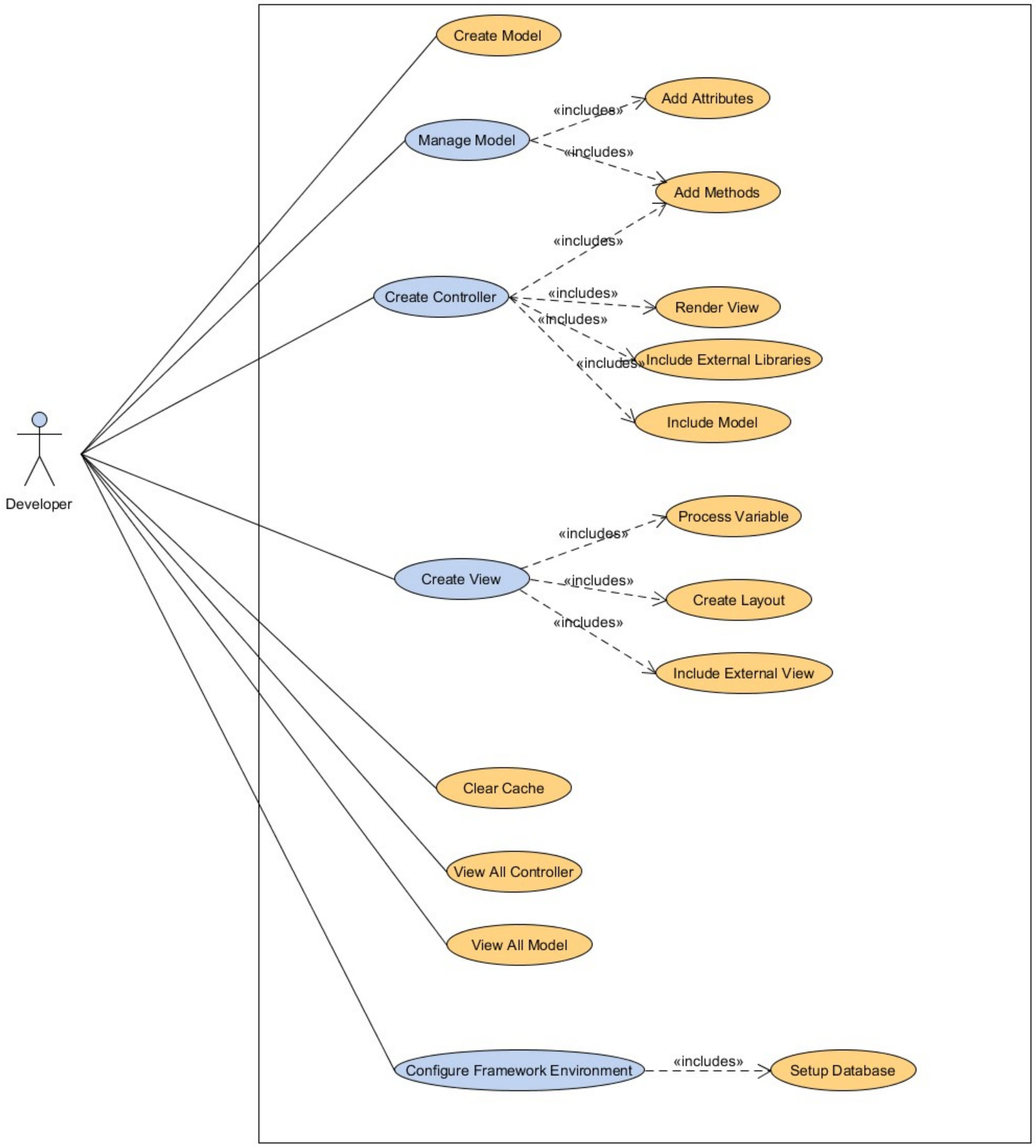
In this chapter software, architecture diagrams are applied to the system. Bovidae is a full featured web-based system that will facilitate developer. The software will provide a solution to the developer.

4.2 Architectural Design:

In Bovidae framework architecture, we describe its group of components, their connections, interactions among them and deployment configuration of all components. Data Tier layer of Bovidae framework includes queries, indexting and storage. Data Tier is a Database layer of Bovidae. In Data Access Tier Bovidae have interface with DB that handle all data I/O usually statements. This tier has ORM (object relational mapping). In Bussiness Tier of Bovidae, it deals with business objects, rules, data manipulation and transformation into information. In Presentation Logic Tier Bovidae provides business objects that displayed to users of the software. It works thorugh PHP. So Business and Presentation Logic Tier of Bovidae have here controller layer. And in presentation GUI it deals with end user system like html. It handles the view layer of Bovidae framework.



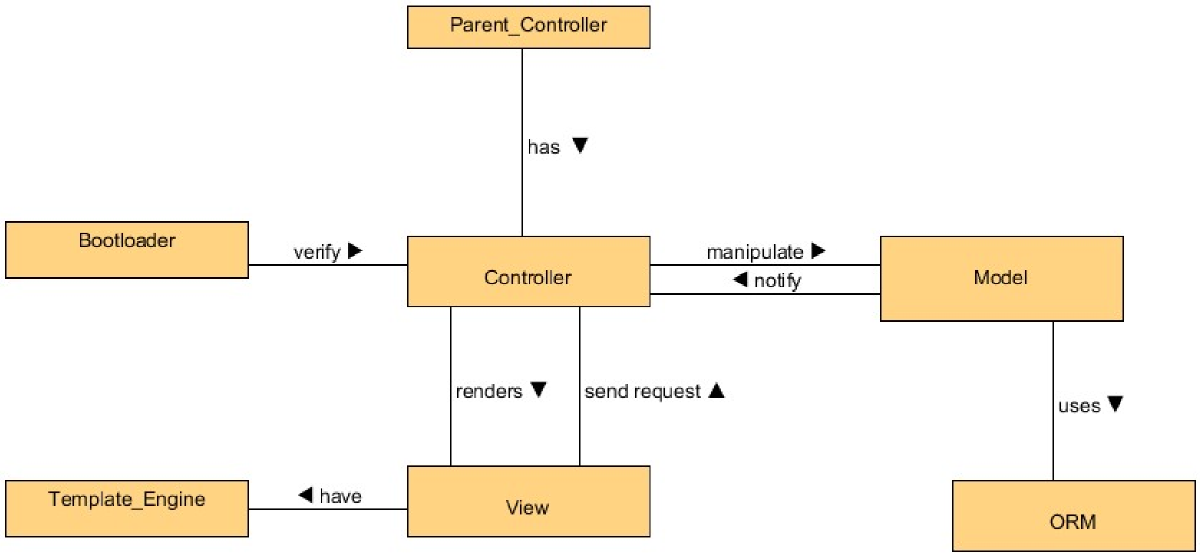
4.3 Use Case Diagram



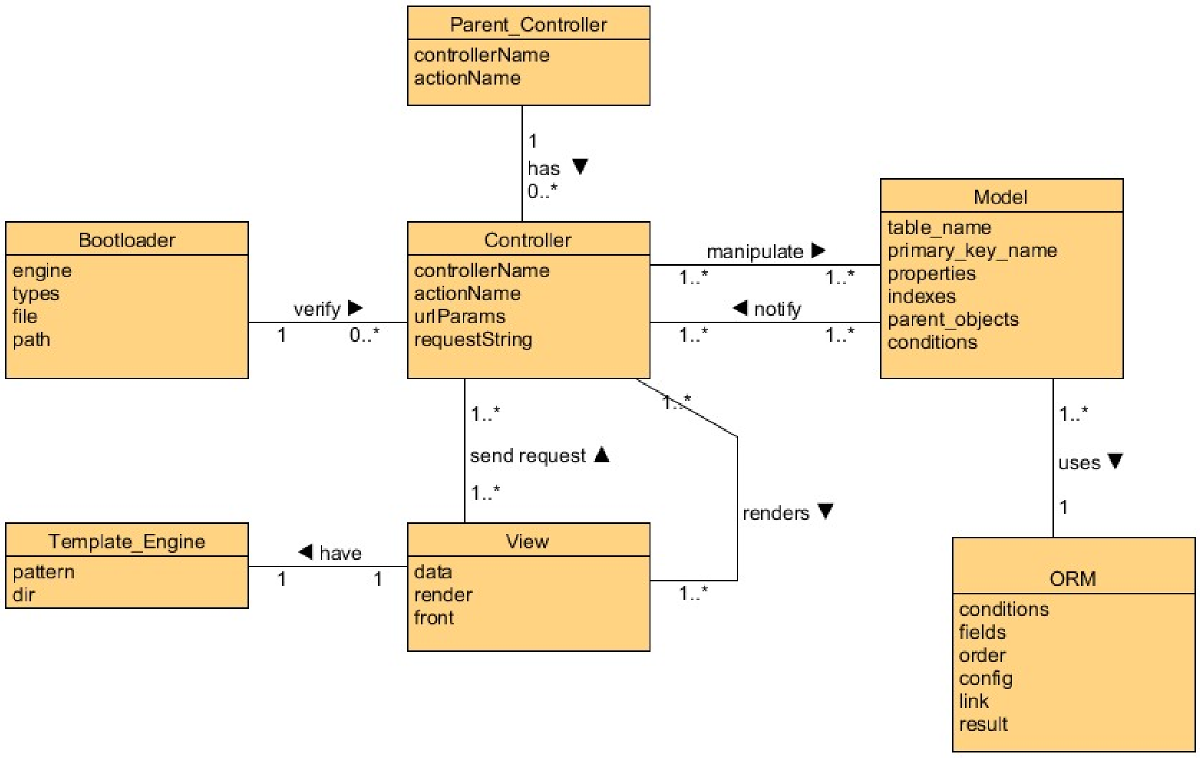
See in Appendix B

4.4 Detailed Design

4.4.1 Domain Model



4.4.2 Entity relationship diagram:



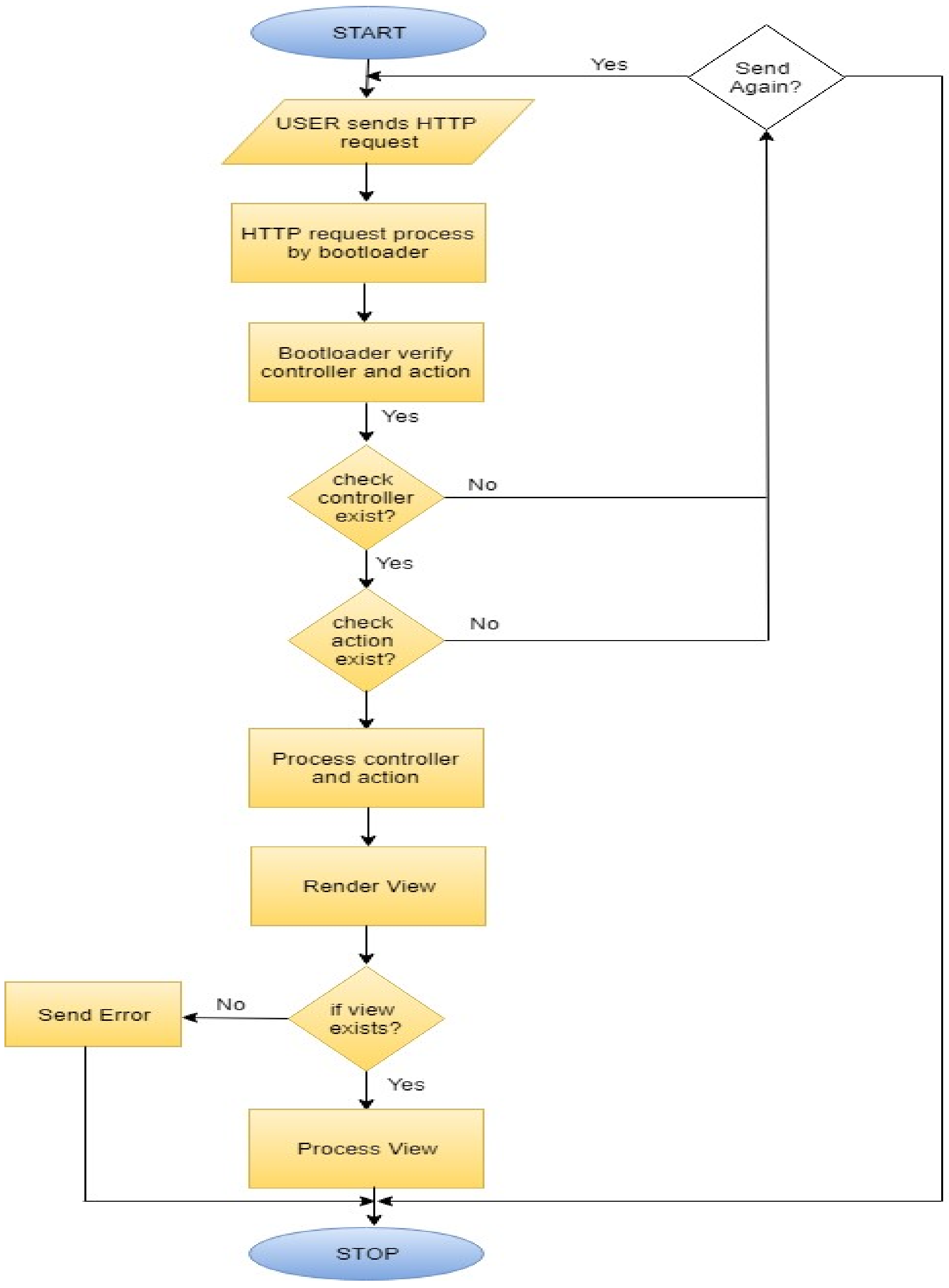
Chapter 5: Implementation

Implementation

5.1 Flow Control/ Pseudocodes

5.1.1 Flow Chart

5.1.1.1 Processing Request



5.2 Components, Libraries, Web Services and stubs

A component in software development is a generic term which can be used purposely according to the software is developing. Components sometimes refer to the hardware components that are required by the software being developing and can also stand for the software components required. We can think of a component as the subset of a module or in other words, one module can have multiple components inside it.

Libraries are the pre-written set of classes, written in some programming language we will be using to assist our development. They will help us in achieving some of our desired functionalities (relevant to graphical UI) without explicitly writing their code.

The choice of using any library depends upon the level of advantages it provides as compared to other similar ones.

5.2.1 Bootstrap

We are using bootstrap for web development. Bootstrap is the most powerful HTML, CSS and JavaScript front-end framework for developing responsive websites. It has a number of UI components for creating a site UI, most of which are responsive themselves.

5.2.2 Language, Framework and Platform

The technologies used for this project are:

- HTML5
- CSS3
- PHP
- JavaScript
- JQuery
- MySQL
- AJAX
- XAMPP

- WAMP

5.3 Best Practices / Coding Standards

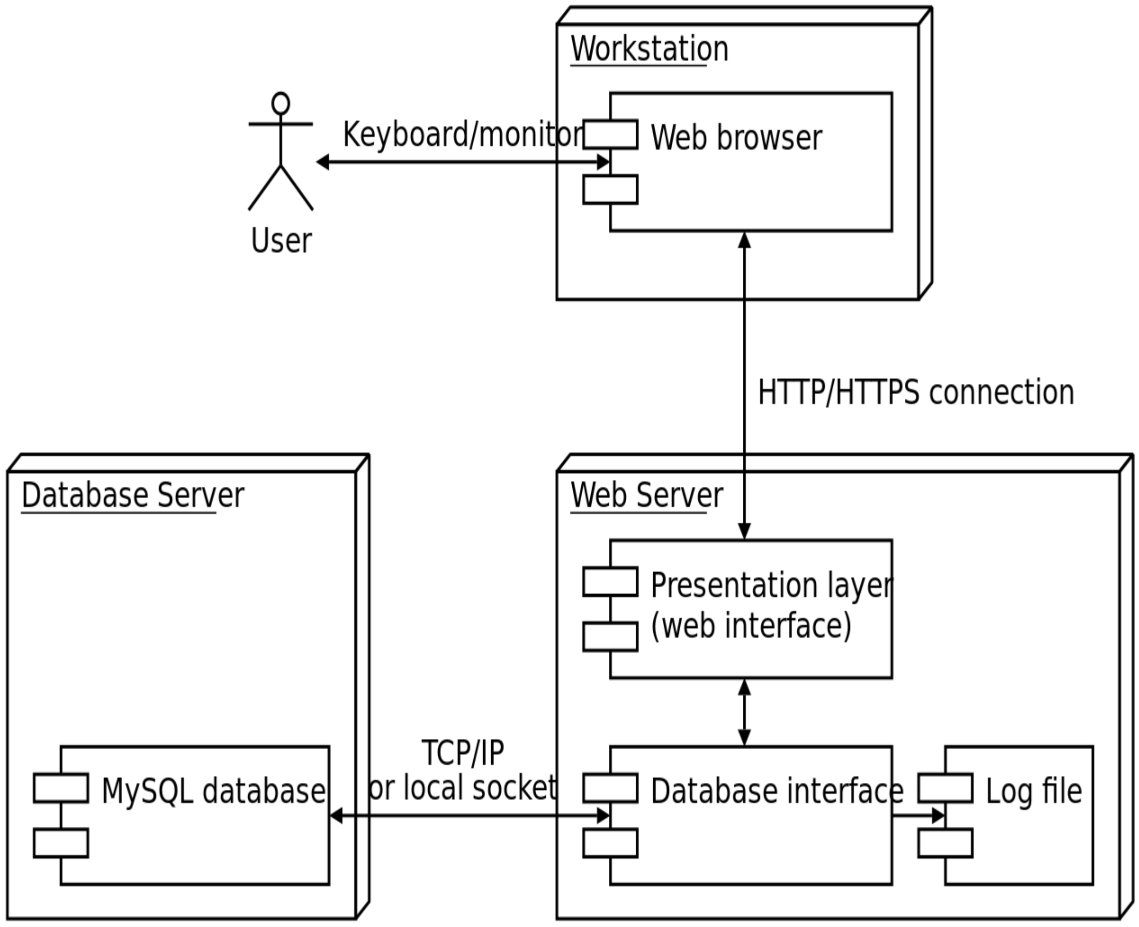
The following conventions are followed:

1. Naming convention

    a. Pages: All in lowercase with dashed separators.

    b. Variable: Follow camel case.

5.4 Deployment Environment



5.5 Summary

This chapter summarizes the details of working of our system, the technologies we are using, a high level overview of different components, libraries.

Chapter 6:
Testing and Evaluation

Testing and Evaluation

6.1 Introduction

Software testing is a process of executing a program or application with the intent of finding the software bugs and to check whether the actual results match the expected results ensuring a defect free software system. The technique followed to test 'Bovidae Framework' system is Black boxing. Black box testing is a method of software testing that examines the functionality of an application based on the specifications and is also known as Specifications based testing. Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

6.2 List of Test Scenarios

Due to time constraints, it was not possible to apply black box testing on the whole system. The lists of test scenarios considered for the project are as follows:

See in Appendix C

6.3 Performance and Evaluation:

The test cases are written were performed to evaluate whether the system conforms to the intended functionality. The results showed that all the test cases performed were passed and that the actual behavior matched the expected behavior.

6.4 Summary:

In this chapter, we discussed all the test cases performed on the system and the importance of testing, core functional requirements, and evaluation of some major functional requirements of the system.

Chapter 7:
Conclusion and Outlook

Conclusion and Outlook

7.1 Introduction:

In this chapter, we would like to include the achievements and the lessons learned while developing the system, and how we faced and solved this project by first doing extensive research, requirement gathering and then finding multiple solutions to it and using an optimal solution that fits as well as improvements which we would like to include in future.

7.2 Achievements and Improvements:

With the grace of Allah Almighty, we were able to develop the targeted features for the Bovidae Framework; it's an easy way to build full-featured web applications. The main achievement is the practical implementation of this project and learning how to work in a team. We worked together in every hurdle that we faced during the implementation of this project. We have improved our concepts, knowledge, and skills of database and web development. We have learned how to handle requirement changes, practically and satisfy our customer. After the development of this project, we are capable of building any similar product. For us, the biggest achievement is that we have developed a system that will help developers who face a lot of difficulties when they are working on web projects. With the completion of this project, we provided the latest framework in the way of PHP and we are so happy with this achievement.

7.3 Critical Review:

Bovidae Framework is an open-source framework for developing web applications, using object-oriented code. Bovidae is a powerful PHP framework, built for PHP coders who need a simple way to create full-featured web applications. Its goal is to enable you to develop projects much faster than you could. Bovidae lets you creatively focus on your project by minimizing the amount of code needed for a given task. The main idea behind this project is to give relief to PHP developers and give a framework with less file size. It will give a platform to developers who face a lot of problems while working on projects.

7.4 Future recommendation /outlook

We have developed Bovidae Framework with best of our efforts and within given time constraints. Due to time limitation, we were unable to implement some of the features. Due to human nature, there is always a need for further Improvement and feature additions. In the future, we will continue to improve our system. Next version of Bovidae Framework will include these features:

- SQL and PostgreSQL

7.5 Summary:

This chapter concludes the project report by looking into the improvement and achievement made during the development. This chapter consists of our achievements and improvements, critical review of application, limitations and future recommendations. Bovidae Framework will be a powerful PHP framework which is built for developers who need a simple way to create full-featured web applications. Bovidae Framework is Simple solutions over complexity. The Bovidae framework also adds structure to the code, prompting the developer to write better, more readable, and more maintainable code. Ultimately, this framework makes programming easier, since it packages complex operations into simple statements.

Reference and Bibliography

[1] https://www.quora.com/What-are-the-advantages-of-using-PHP
[2] https://www.jotform.com/blog/discussing-php-frameworks/
[3] https://www.jotform.com/blog/discussing-php-frameworks/
[4] http://socialcompare.com/en/comparison/php-frameworks-comparison
[5] https://www.grossum.com/blog/php-frameworks-comparison-discover-top-10-best-php-frameworks-in-2017
[6] https://en.wikipedia.org/wiki/Composer_(software)

Appendix

Appendix A:

Software Requirements Specifications (SRS)

FR 3.3.1 Developer:

3.4.3.1	Developer shall be able to create a Model.
3.4.3.2	Developer shall be able to create a controller.
3.4.3.3	Developer shall be able to create a view.
3.4.3.4	Developer shall be able to Add Attributes.
3.4.3.5	Developer shall be able to Add Methods.
3.4.3.6	Developer shall be able to manage model.
3.4.3.7	Developer shall be able to include external libraries.
3.4.3.8	Developer shall be able to include models.
3.4.3.9	Developer shall be able to render the view.
3.4.3.10	Developer shall be able to create layouts.
3.4.3.11	Developer shall be able to set database.
3.4.3.12	Developer shall be able to clear cache.
3.4.3.13	Developer shall be able to view all model.
3.4.3.14	Developer shall be able to view all controllers.
3.4.3.15	Developer shall be able to configure the framework environment.
3.4.3.16	Developer shall be able to create an external view.

Appendix B:

4.4.3 Use Case Specification

4.4.3.1 Fully Dressed Form

First, we will give a description of the actors in the Bovidae domain who will going to use the system.

ACTOR	GOALS
Developer	The **Developer** work on Creating Model, Creating Controllers, Creating View, Rendering View, Including Model, Creating Layouts and etc.

4.4.3.2 Use Cases:

1. Create Model
2. Manage Model
3. Add Attributes
4. Add Methods
5. Create Controller
6. Render View
7. Include External Libraries
8. Include Model
9. Create View
10. Process Variable
11. Create Layout
12. Include External View
13. View All Controller
14. View All Model
15. Configure Framework Environment
16. Setup Database

4.4.3.2.1 USE CASE UC1: Create Model

Description:

Developers will Create Model on the system.

UC_ID	UC_01
Use case name	Create Model
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created Controller.
Post-Condition	The actor is now able to work on the model. If not the system state is unchanged.
Main Success Scenario	1. The developer will initiate a request of creating a model on the system. 2. The system will show model area to the developer. 3. The system will ask for adding table and model name explicitly. 4. The developer will accept. 5. The developer will write table name on the system 6. The developer will write the model name on the system. 7. The system will create the model. 8. If the developer will not select table and model name explicitly system will ask for select table name from the

	existing model. 9. The developer will select the table on the system. 10. The system will show the model name automatically. 11. The system will create the model. 12. The developer will end process.
Alternative flow	2a. The system will show error and will not show model. 3a. The system will not ask for adding table and model name explicitly. 5a. the developer will not enter a table name. 5b. the developer will add the wrong name of the table. 5c. the developer will enter the outstretched name of the table. 6a. the developer will not enter a model name. 6b. the developer will enter the wrong name of the model. 6c. the developer will enter the outstretched name of the model. 7a. The system will not create the model. 7b. The system will create a file of the model but not write any data in the file. 8a. The system will not show the existing table name. 9a. The developer will not select a table name. 10a. The system will not show the model name

	automatically.
Special Requirement	None

4.4.3.2.2 USE CASE UC2: Manage Model

Description:

Developers will Manage Model on system.

UC_ID	UC_02
Use case name	Manage Model
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created Controller. 3. The developer must have created the model.
Post-Condition	The actor is now able to see updated content of the model. The system state is unchanged.
Main Success scenario	1. Developer will request for manage model. 2. The developer will ask for update model. 3. System gives access for updating model 4. The developer will update the model. 5. The system will confirm model is updated. 6. The developer will end process.
Alternative flow	None

Special Requirement	None

4.4.3.2.3 USE CASE UC3: Add Attributes

Description:

The developer will add attributes in Model in the system.

UC_ID	UC_03
Use case name	Add Attributes
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. Developers must have created Controller. 3. Developers must have created the model.
Post-Condition	The actor is now able to see newly added attributes in the model. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for adding new attributes in the model to the system. 2. The system will accept a developer request. 3. The developer will add new attributes. 4. The system will authenticate attributes. 5. The system will confirm attributes. 6. The system will display message attributes are added. 7. The developer will end process.
Alternative flow	1a. The developer will not add attributes to the model.

	2a. The System will not respond to the developer. 2b. The System will not accept a developer request.
Special Requirement	None

4.4.3.2.4 USE CASE UC4: Create Controller

Description:

Developers will Create Controller in Controller on system.

UC_ID	UC_04
Use case name	Create Controller
Primary Actor	Developer
Pre-Conditions	1. Developers must have created a view. 2. Developers must have created model.
Post-Condition	The actor is now able to work on a controller. If not the system state is unchanged.
Main Success Scenario	1. The developer will initiate a request of creating a controller to the system. 2. The system will show the controller area to the developer. 3. System asks to developer of entering controller name on system. 4. Developer will enter controller name. 5. The system will ask for the select model from existing models. 6. Developer will select model name.

	7. System will create controller. 8. The system will display the message controller is created. 9. Developer will end process.
Alternative flow	1a. the developer will not create a controller. 2a. The system will not show controller area to the developer. 2b. The system will not respond to the developer. 3a. The system will not ask developer for entering controller name. 3b. The system will show error. 4a. The developer will enter wrong controller name. 4b. The developer will not enter the controller name. 5a. The system will no ask developer for selecting a model name. 6a. the developer will not select a model name. 7a. The system will not create a controller. 8a. the system will not show any message. 8a. The system will create a controller but not show any displayed message.
Special Requirement	None

4.4.3.2.5 USE CASE UC5: Render View

<u>Description:</u>

The developer will render View for Controller on the system.

UC_ID	UC_05

Use case name	Render View
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The use case was successful, the actor is now able to link with a view. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for rendering view to the system. 2. The system will accept a developer request. 3. The system will render view for the controller. 4. The system will display message view is rendered. 5. The developer will end process.
Alternative flow	3a. the system will not render view for the controller. 3b. The system will render view but not showing that view is rendered.
Special Requirement	None

4.4.3.2.6 USE CASE UC6: Add Methods

Description:

The developer will add Methods for Controller and model on the system.

UC_ID	UC_06
Use case name	Add Methods

Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to work on a controller. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for adding methods in the controller to the system. 2. The system will accept a developer request. 3. The developer will add methods. 4. The system will authenticate methods. 5. The system will confirm methods. 6. The system will display message methods are added 7. The developer will end process.
Alternative flow	3a. The developer will not add methods. 5a. The system will not confirm methods.
Special Requirement	None.

4.4.3.2.7 USE CASE UC7: Include External Libraries

Description:

Developers will Include External Libraries for Controller on the system.

UC_ID	UC_07

Use case name	Include External Libraries
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to work on a controller. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for including external libraries in the controller to the system. 2. The system will accept a developer request. 3. The system will include libraries. 4. The system will display message libraries are included. 5. The developer will end process.
Alternative flow	1a. The developer will not add external libraries to the controller 3a. The system doesn't connect libraries to the controller. 3b. The system will not show external libraries. 4a. System state will remain the same.
Special Requirement	None

4.4.3.2.8 USE CASE UC08: Include Model

Description:

Developers will Include Model for Controller on the system.

UC_ID	UC_08
Use case name	Include Model
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to link with the model. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for include model to the controller to the system. 2. The system will accept a developer request. 3. The system will include the model. 4. The system will display the message model is included. 5. The developer will end process.
Alternative flow	1a. The developer will not include a model to the controller 3a. The system will not include the model. 4a. System state will remain the same.
Special Requirement	None

4.4.3.2.9 USE CASE UC09: Create View

Description:

Developers will Create View on the system.

UC_ID	UC_09
Use case name	Create View
Primary Actor	Developer
Pre-Conditions	1. The developer must have created Controller.
Post-Condition	The actor is now able to see the view. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for creating a view. 2. System show view area to the developer. 3. The system will ask for the path to the developer. 4. The developer will enter their desired path on the system. 5. The system will ask for entering the name of the view file. 6. The developer will enter the name of view file. 7. The system will create a view. 8. The system will display message view is created. 9. The developer will end process.
Alternative flow	4a. Developer enters the wrong path. 4b. The developer will not enter the path. 5a. The system will not ask for entering view name. 6a. The developer will enter the wrong name of the file. 6b. The developer will not enter view name. 7a. The system will not create view.
Special Requirement	None

4.4.3.2.10 USE CASE UC10:Process Variable

Description:

Developers will Process Variable on View on system.

UC_ID	UC_10
Use case name	Process Variable
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to see changes in the controller. If not the system state is unchanged.
Main Success scenario	1. Developer will request for Process Variable. 2. The system will give access for Process Variable. 3. The system will ask for entering variables. 4. The developer will enter variables. 5. The system will display message variables are processed. 6. The developer will end process.
Alternative flow	None
Special Requirement	None

4.4.3.2.11 USE CASE UC11: Create Layout

Description:

Developers will Create Layout in View.

UC_ID	UC_11
Use case name	Create a Layout
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to see the interface of view. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for creating layouts. 2. The system will give access for creating layouts. 3. The system will ask for selecting layouts. 4. The developer will select layouts. 5. The developer will end process.
Alternative flow	None
Special Requirement	None

4.4.3.2.12 USE CASE UC12: Include External View

Description:

Developers will Include External View in View.

UC_ID	UC_12
Use case name	Include External View

Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created a model. 3. The developer must have created Controller.
Post-Condition	The actor is now able to see the interface of view. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for Including External View. 2. The system will give access to Including External View. 3. The system will ask for selecting your external view from your device. 4. The developer will select the external view. 5. The system will display message external view is included. 6. The developer will end process.
Alternative flow	1a. The developer will not add external view. 3a. The system will not show external views. 5a. System state will remain the same.
Special Requirement	None

4.4.3.2.13 USE CASE UC14: View All Controller

Description:

Developers will View All Controller on the system.

UC_ID	UC_13

Use case name	View All Controller
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor will be able to see all controllers. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for viewing all controllers. 2. The system gives access to the developer. 3. The system will display all controllers to developers. 4. The developer will end process.
Alternative flow	3a. The system doesn't show a list of controllers 3b. The system will ask for refresh
Special Requirement	None

4.4.3.2.14 USE CASE UC15: View All Model

Description:

Developers will View All Model on the system.

UC_ID	UC_14
Use case name	View All Model
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view.

	2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor will be able to see all the models. If not the system state is unchanged.
Main Success Scenario	1. The developer will request for viewing all models. 2. The system gives access to the developer. 3. The system will display all models to developers. 4. The developer will end process.
Alternative flow	3a. The system will not show a list of models 3b. The system will ask for refresh
Special Requirement	None

4.4.3.2.15 USE CASE UC16: Configure Framework Environment

Description:

Developers will Configure Framework Environment on the system.

UC_ID	UC_15
Use case name	Configure Framework Environment
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model. 3. The developer must have created Controller.
Post-Condition	The actor will be able to see all attached databases. If not the system

	state is unchanged.
Main Success Scenario	1. The developer will request for Configure Framework Environment. 2. The system gives access to the developer. 3. The system will ask developer for selecting their desired databases. 4. The developer will select their desired database. 5. The system will connect project model with the database. 6. The system will display message successfully connected. 7. The developer will end process.
Alternative flow	3a. The system will not show any database. 4a. The developer will not select any database.
Special Requirement	None

4.4.3.2.16 USE CASE UC17: Setup Database

Description:

Developers will Setup Database on the system.

UC_ID	UC_16
Use case name	Setup Database
Primary Actor	Developer
Pre-Conditions	1. The developer must have created a view. 2. The developer must have created the model.

	3. The developer must have created Controller.
Post-Condition	The actor will be able to see all attached databases. If not the system state is unchanged.
Main Success scenario	1. Developer will initiate request of Setup Database. 2. The system gives access to the developer. 3. The system will display all setup databases. 4. The developer will end process.
Alternative flow	Non
Special Requirement	None

Appendix C: List of Test Scenarios

[TC- 1] Create Model

Test Case #:	TC – 1
System:	Bovidae Framework
Test Case Name:	Create Model
Related Requirement:	FR-3.3.1.1
Short Description:	Test that developer can create a model
Designed Date:	29/11/2018
Execution Date:	4/12/2018

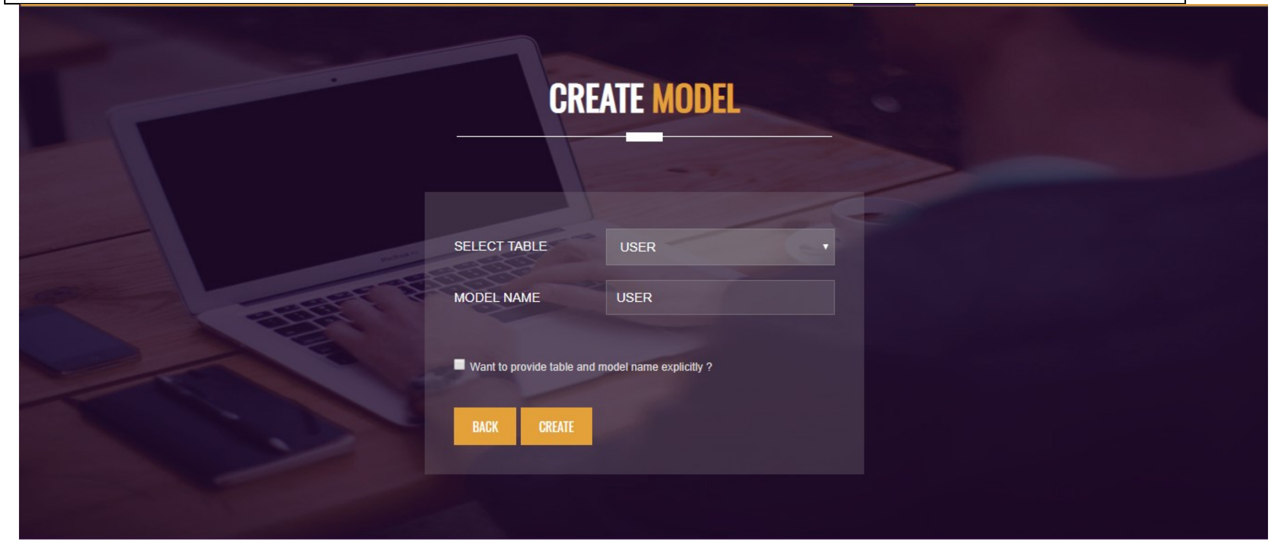
Pre-Condition:

1. The developer has created a view.
2. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of Create Model Page (http://localhost/Bovidae/public/generator/create_model)	The system should display Create Model page.	The system displays Create Model page.	Pass
2	The developer selects Table from Create Model form.	The system should display tables from Database and developer select table from them.	The system display tables from database and developer select table from these tables.	Pass
3	System enter model name automatically.	The system should show Model name automatically.	The system show model name	Pass

			automatically.	
4	The developer selects table and model name explicitly.	The system should show table name and model name fields to the developer so he can enter data in these fields.	System show table name and model name fields to the developer so he can enter data in these fields.	Pass
5	Developer Click on Create Model button	The system should create Model	The system creates model	Pass

Post-Condition: The model class is created.



SELECT TABLE USER

MODEL NAME USER

☐ Want to provide table and model name explicitly ?

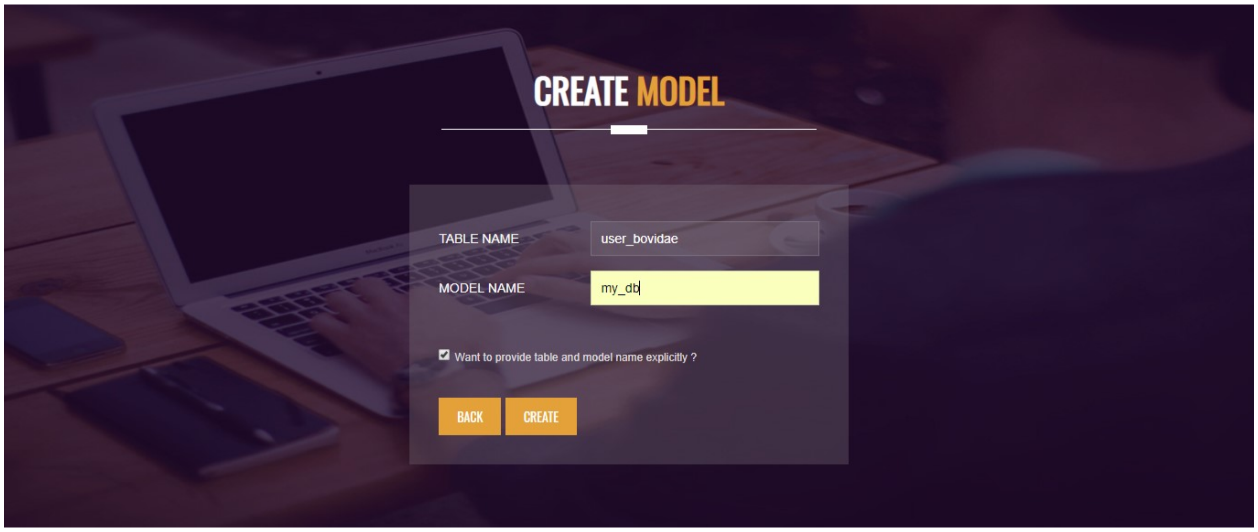


TABLE NAME user_bovidae

MODEL NAME my_db

☑ Want to provide table and model name explicitly ?

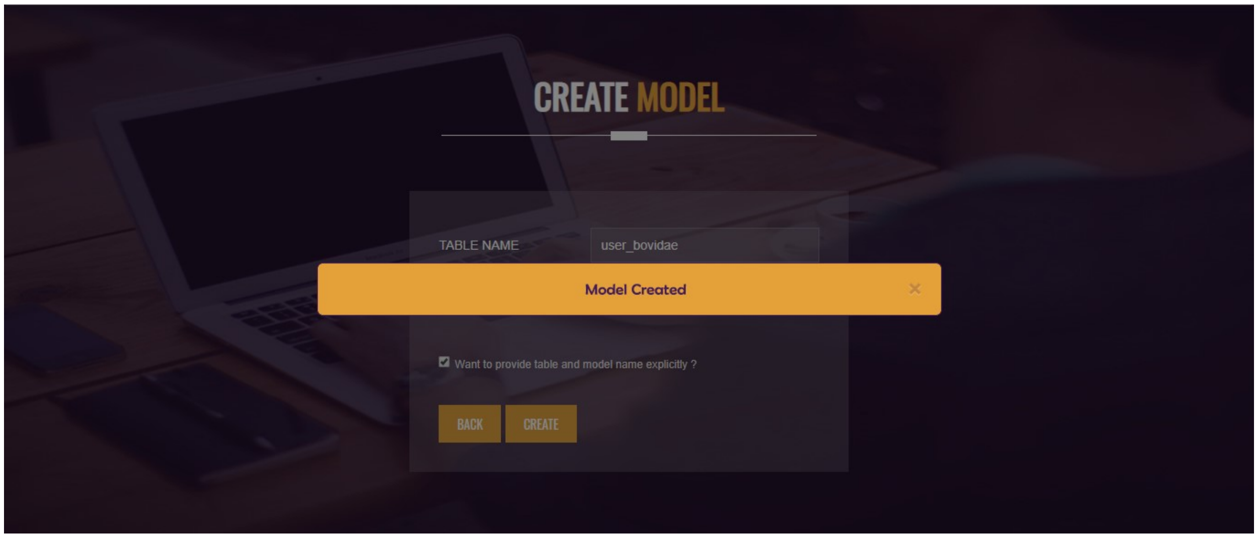


TABLE NAME user_bovidae

Model Created

☑ Want to provide table and model name explicitly ?

[TC- 2] Create Controller

Test Case #:	TC – 2
System:	Bovidae Framework
Test Case Name:	Create Controller
Related Requirement:	FR-3.3.1.19
Short Description:	Test that developer can create controller
Designed Date:	29/11/2018
Execution Date:	4/12/2018

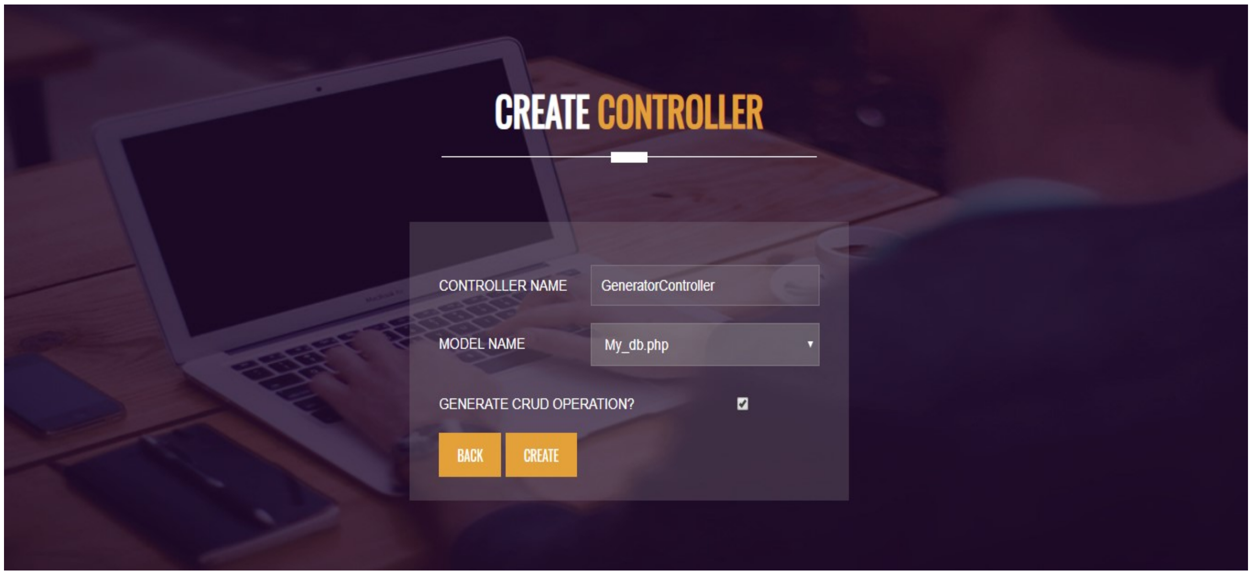
Pre-Condition:

1. The developer has created a view.
2. The developer has created a Model.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of Create controller Page (http://localhost/Bovidae/public/generator/create_controller)	The system should display Create Controller page.	The system displays Create Controller page.	pass
2	Developer enters a controller name.	The system should create controller class with that name.	The system creates controller class with that name.	pass
3	The developer selects getter and setter from the form.	The system should generate getter and setter in file.	The system generates getter and setter in file.	pass
4	If Developer doesn't select getter and setter from a form.	The system should not generate getter and setter in file.	The system doesn't generate getter and setter in file.	pass
5	Developer Click on Create Controller button	The system should create Controller	The system creates Controller	pass

Post-Condition:

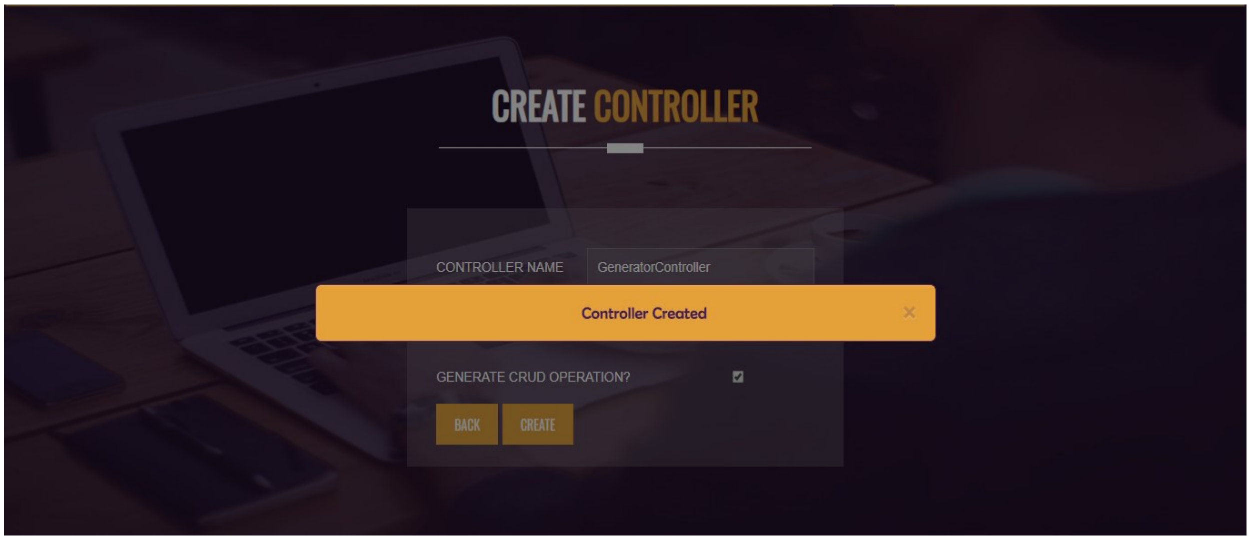
1. The Controller class is created.



CONTROLLER NAME GeneratorController

MODEL NAME My_db.php

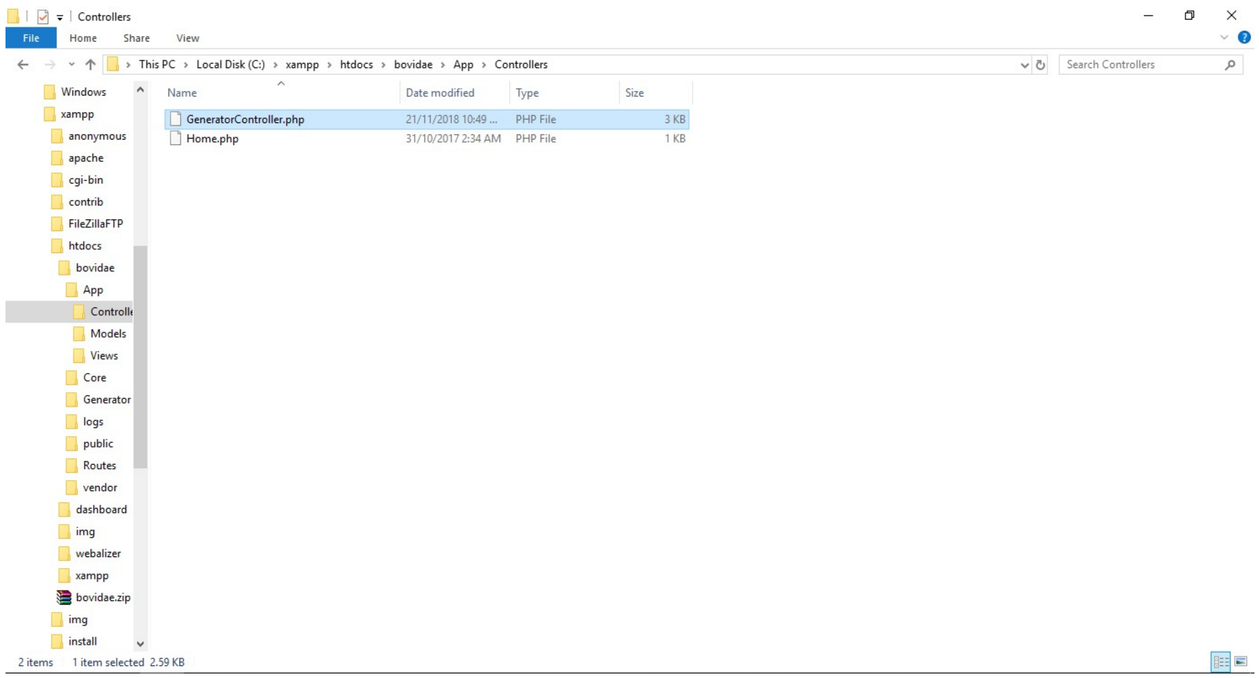
GENERATE CRUD OPERATION? ☑



CONTROLLER NAME GeneratorController

Controller Created

GENERATE CRUD OPERATION? ☑



Name	Date modified	Type	Size
GeneratorController.php	21/11/2018 10:49 ...	PHP File	3 KB
Home.php	31/10/2017 2:34 AM	PHP File	1 KB

[TC- 3] Create View

Test Case #:	TC – 3
System:	Bovidae Framework
Test Case Name:	Create View
Related Requirement:	FR-3.3.1.17
Short Description:	Test that developer can create their view
Designed Date:	29/11/2018
Execution Date:	4/12/2018

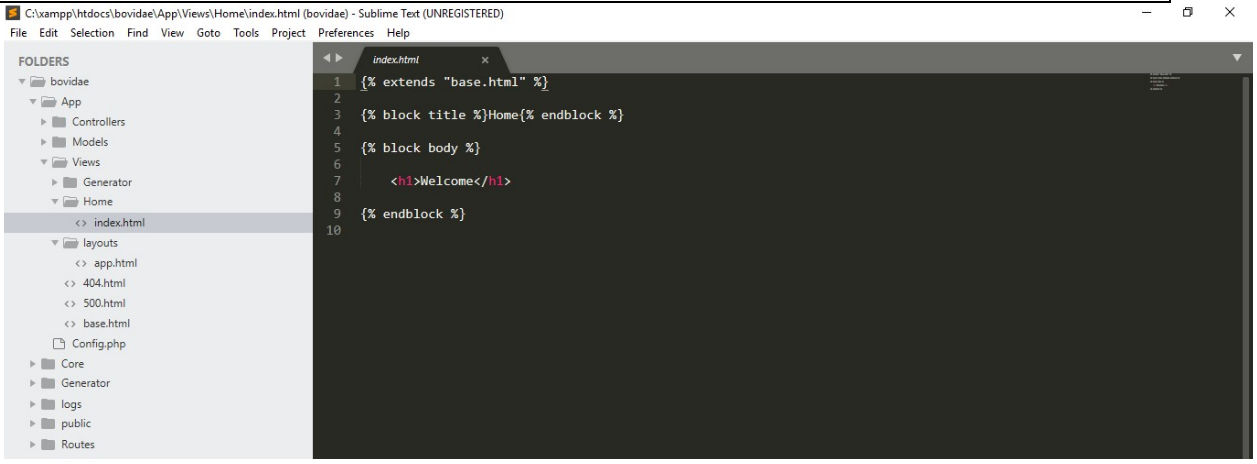
Pre-Condition:

1. The developer has created Model.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of Create controller Page (http://localhost/Bovidae/publi	The system should display Create View page.	The system displays Create View page.	pass

	c/generator/create_view			
2	Developer enters Path and name of view.	The system should take path and name of view.	The system takes path and name of view.	Pass
3	Developer Click on Create View button	The system should create View	The system creates View	Pass

Post-Condition: The view is created.


C:\xampp\htdocs\bovidae\App\Views\Home\index.html (bovidae) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
bovidae
App
Controllers
Models
Views
Generator
Home
index.html
layouts
app.html
404.html
500.html
base.html
Config.php
Core
Generator
logs
public
Routes
index.html

```
{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block body %}

    <h1>Welcome</h1>

{% endblock %}

```

[TC- 5] View All Model

Test Case #:	TC – 5
System:	Bovidae Framework
Test Case Name:	View All Model
Related Requirement:	FR-3.3.1.14
Short Description:	Test that all model is viewable or not
Designed Date:	29/11/2018
Execution Date:	04/12/2018

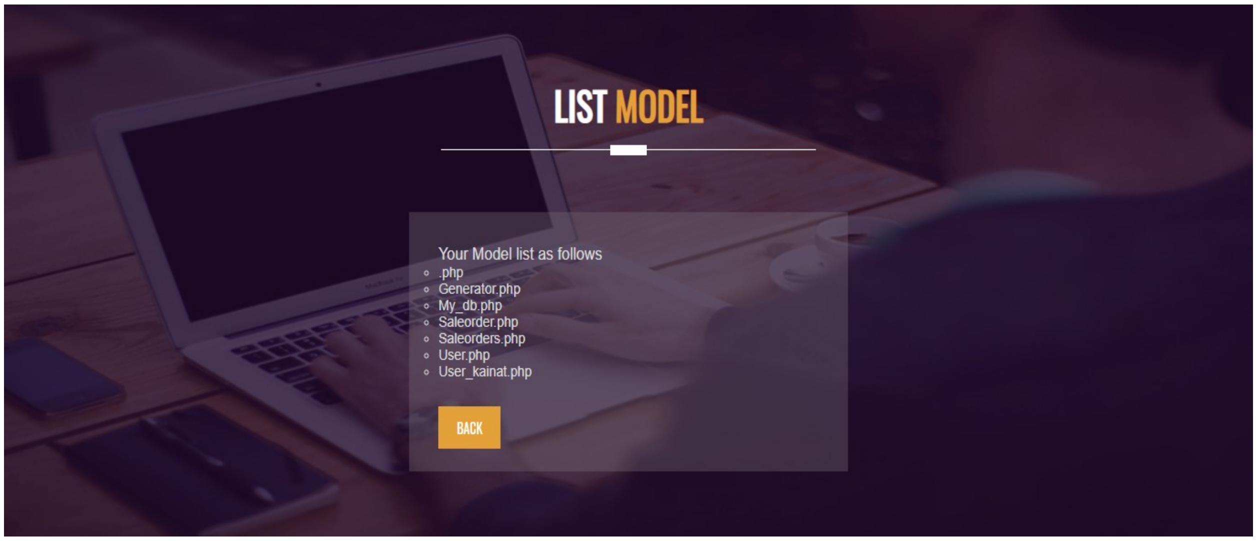
Pre-Condition:

1. The developer has created view.
2. The developer has created model.
3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of view all model Page (http://localhost/Bovidae/public/generator/all_models)	The system should display view all model page.	The system displays view all model pages.	Pass
2	Developer checks the model list.	The system should show all model lists.	The system show all model lists.	Pass

Post-Condition:

1. The developer is able to see all models.



Your Model list as follows
- .php
- Generator.php
- My_db.php
- Saleorder.php
- Saleorders.php
- User.php
- User_kainat.php

[TC- 6] View All Controller

Test Case #:	TC – 6
System:	Bovidae Framework
Test Case Name:	View All controller
Related Requirement:	FR-3.3.1.15
Short Description:	Test that all controller are viewable or not
Designed Date:	29/11/2018
Execution Date:	04/12/2018

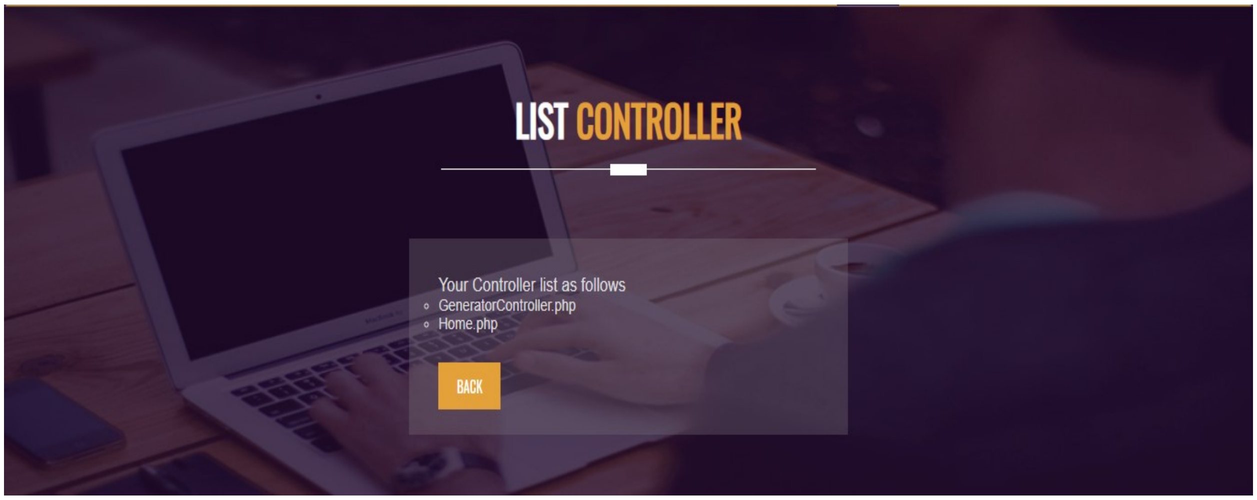
Pre-Condition:

1. The developer has created view.
2. The developer has created model.
3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of view all model Page (http://localhost/Bovidae/public/generator/all_controllers)	The system should display view all model page.	The system displays view all model pages.	Pass
2	Developer checks the controller list.	The system should show all controller lists.	The system shows all controller lists.	Pass

Post-Condition:

1. The developer is able to see all controllers.



Your Controller list as follows

- GeneratorController.php
- Home.php

[TC- 7] Setup Database

Test Case #:	TC – 7
System:	Bovidae Framework
Test Case Name:	Setup Database
Related Requirement:	FR-3.3.1.12
Short Description:	Test that databases are configured
Designed Date:	29/11/2018
Execution Date:	04/12/2018

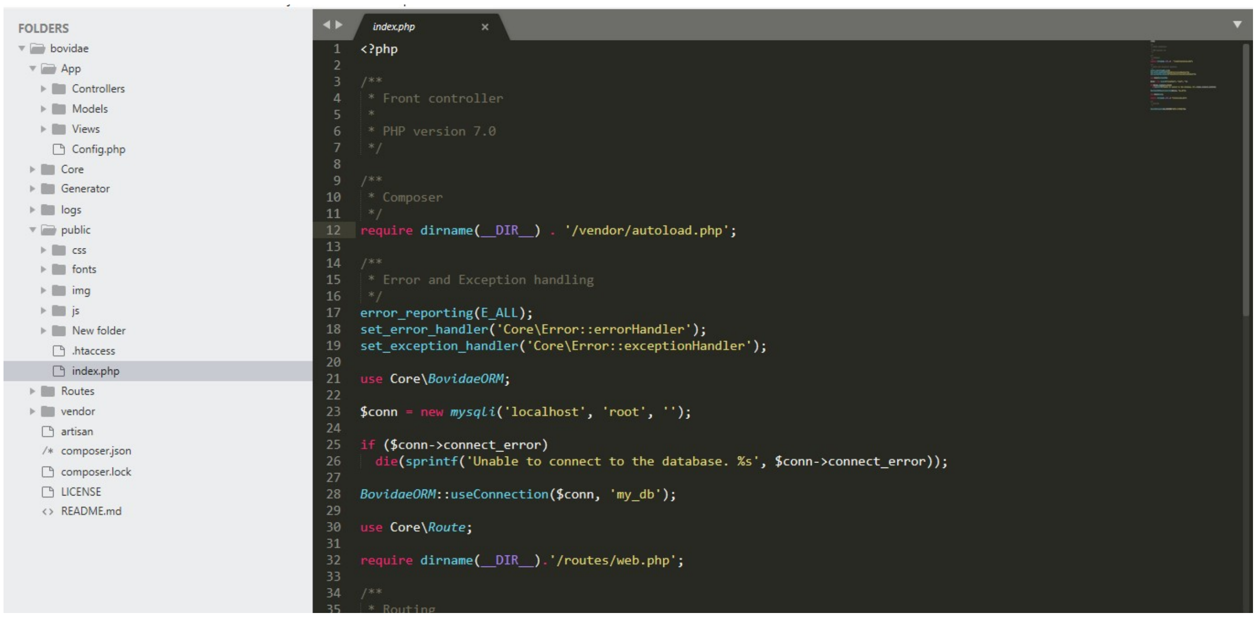
Pre-Condition:

1. The developer has created view.
2. The developer has created model.
3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	The developer selects his desired database from given databases.	The system should display databases and give access to the	The system displays databases and give access to	Pass

		developer to select and work on those databases.	the developer to select and work on that databases.	

Post-Condition:
1. The database is set for the developer's project.



```php
<?php

/**
 * Front controller
 *
 * PHP version 7.0
 */

/**
 * Composer
 */
require dirname(__DIR__) . '/vendor/autoload.php';

/**
 * Error and Exception handling
 */
error_reporting(E_ALL);
set_error_handler('Core\Error::errorHandler');
set_exception_handler('Core\Error::exceptionHandler');

use Core\BovidaeORM;

$conn = new mysqli('localhost', 'root', '');

if ($conn->connect_error)
  die(sprintf('Unable to connect to the database. %s', $conn->connect_error));

BovidaeORM::useConnection($conn, 'my_db');

use Core\Route;

require dirname(__DIR__).'/routes/web.php';

/**
 * Routing
```

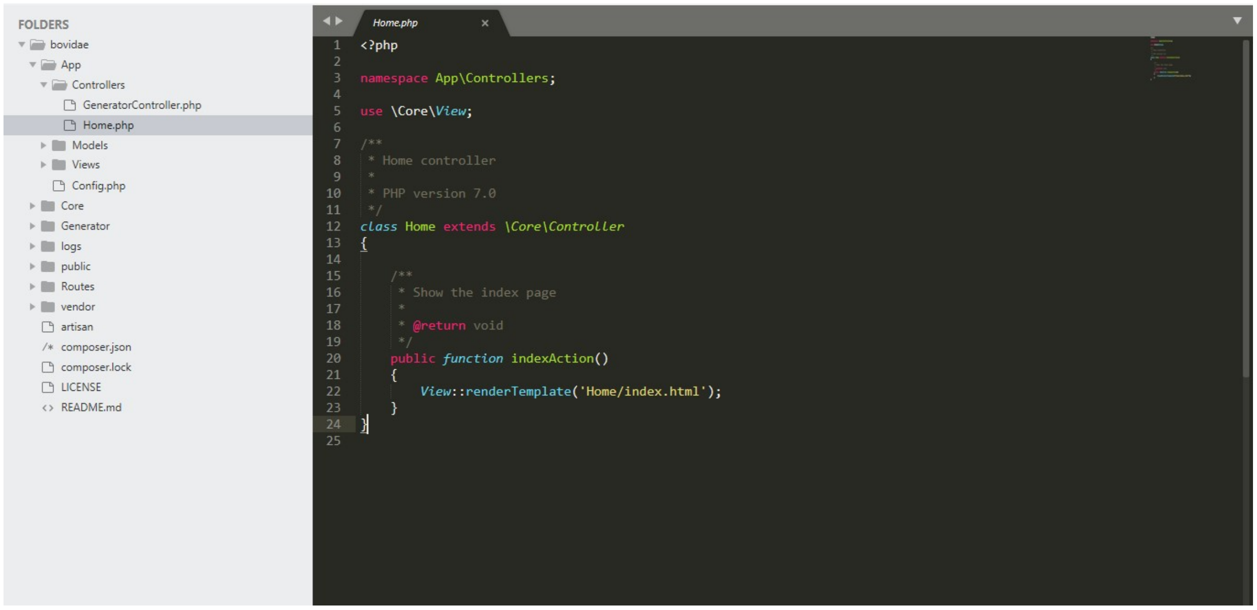
[TC- 8] Add Methods

Test Case #:	TC – 8
System:	Bovidae Framework
Test Case Name:	Add Methods
Related Requirement:	FR-3.3.1.4
Short Description:	Test, that methods are added for model and controller class
Designed Date:	29/11/2018
Execution Date:	04/12/2018

Pre-Condition:
1. The developer has created view. 2. The developer has created model. 3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer give the path of Create controller or create model Page	The system should display Create controller or create model page.	The system displays Create controller or Create model page.	Pass
3	The developer selects getter and setter from form.	The system should generate getter and setter in the file.	The system generates getter and setter in the file.	Pass
4	If Developer doesn't select getter and setter from the form.	The system should not generate getter and setter in the file.	The system doesn't generate getter and setter in the file.	Pass
5	The developer selects constructor option.	The system should create Constructor.	The system creates Constructor	pass

Post-Condition:
1. Methods are added to the controller class and model class.



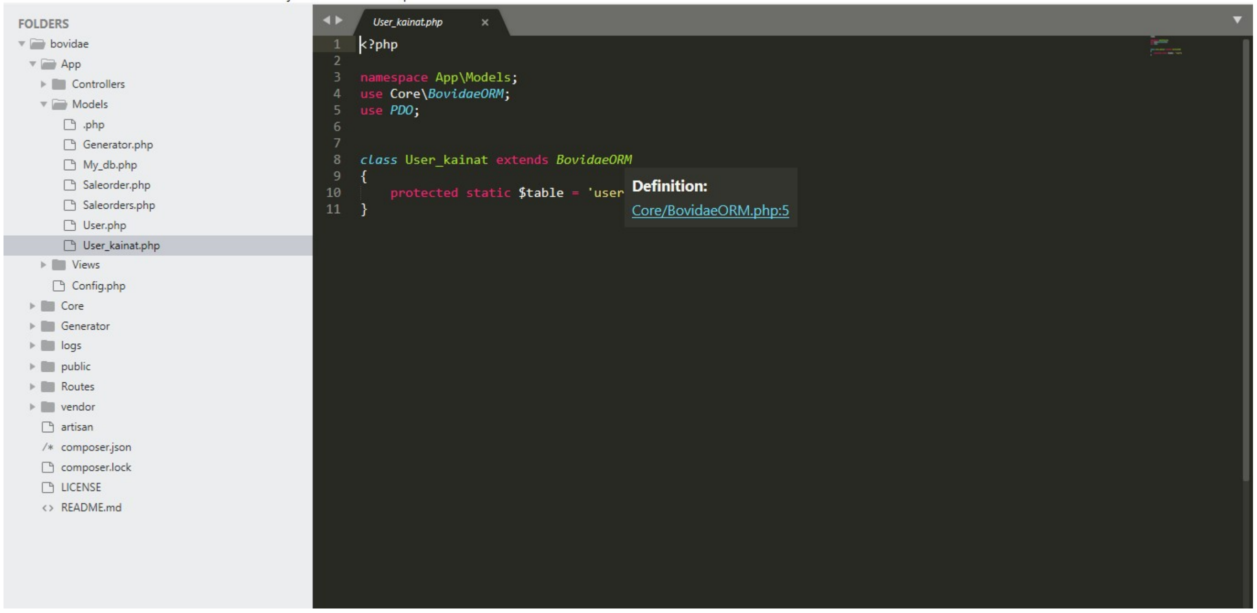
```php
<?php

namespace App\Controllers;

use \Core\View;

/**
 * Home controller
 *
 * PHP version 7.0
 */
class Home extends \Core\Controller
{

    /**
     * Show the index page
     *
     * @return void
     */
    public function indexAction()
    {
        View::renderTemplate('Home/index.html');
    }
}
```



```php
<?php

namespace App\Models;
use Core\BovidaeORM;
use PDO;


class User_kainat extends BovidaeORM
{
    protected static $table = 'user
}
```

[TC- 9] Render View

Test Case #:	TC – 9
System:	Bovidae Framework
Test Case Name:	Render View
Related Requirement:	FR-3.3.1.10

Short Description:	Render View for Controller on system
Designed Date:	29/11/2018
Execution Date:	

Pre-Condition:

1. The developer has created view.
2. The developer has created model.
3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer insert new changes in view	The system should render a view.	The system render view	Pass
2	Developer opens controller	The system should render view for the controller and controller work according to new changes in the view class.	The system renders view for the controller and controller work according to new changes in the view class.	Pass
3	Developer inserts new changes to view but not connected to the controller.	The system should don't render the view and state remain unchanged, no link between controller and view class.	The system doesn't render the view and state remain unchanged, no link between controller and view class.	Pass

Post-Condition:
1. The developer is able to link with a view

[TC- 10] Include external libraries

Test Case #:	TC – 10
System:	Bovidae Framework
Test Case Name:	Include external libraries
Related Requirement:	FR-3.3.1.10
Short Description:	The test is, developer able to Include External Libraries.
Designed Date:	29/11/2018
Execution Date:	04/12/2018

Pre-Condition:
1. The developer has created view. 2. The developer has created model. 3. The developer has created Controller.

Steps	**Action**	**Expected Output**	**Actual output**	**Pass/Fail**
1	Developer insert new libraries	The system should add libraries and also able to link with a controller.	The system adds libraries and also able to link with a controller.	Pass

Post-Condition:
1. The developer is able to work on a controller.

[TC- 11] Create layouts

Test Case #:	TC – 11
System:	Bovidae Framework
Test Case Name:	Create layouts
Related Requirement:	FR-3.3.1.10
Short Description:	The test is, developer able to create layout in view.
Designed Date:	29/11/2018
Execution Date:	

Pre-Condition:

1. Developer has created view.
2. The developer has created model.
3. The developer has created Controller.

Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer create a layout in view	The system should create layout in view.	The system create layout in view.	Pass
2	Developer create another layout in view	The system should create and handle multiple layouts in view.	The system creates and handles multiple layouts in view.	Pass

Post-Condition:

1. The developer is able to see the interface of view.

[TC- 12] Include external view

Test Case #:	TC – 12
System:	Bovidae Framework
Test Case Name:	Create external view
Related Requirement:	FR-3.3.1.10
Short Description:	The test is, developer able to add external view in view
Designed Date:	29/11/2018
Execution Date:	

Pre-Condition:

1. Developer has created view.
2. The developer has created model.
3. The developer has created Controller.

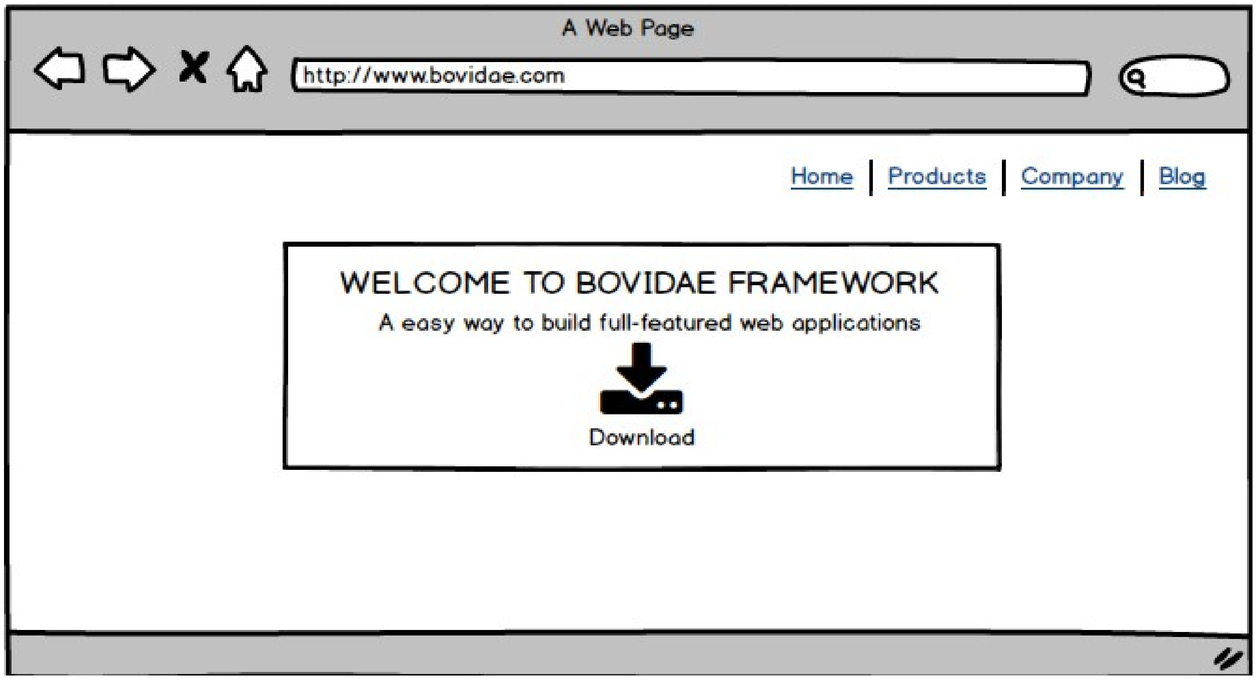
Steps	Action	Expected Output	Actual output	Pass/Fail
1	Developer insert external view	The system should include that external view in view.	The system includes that external view in view.	Pass

Post-Condition:

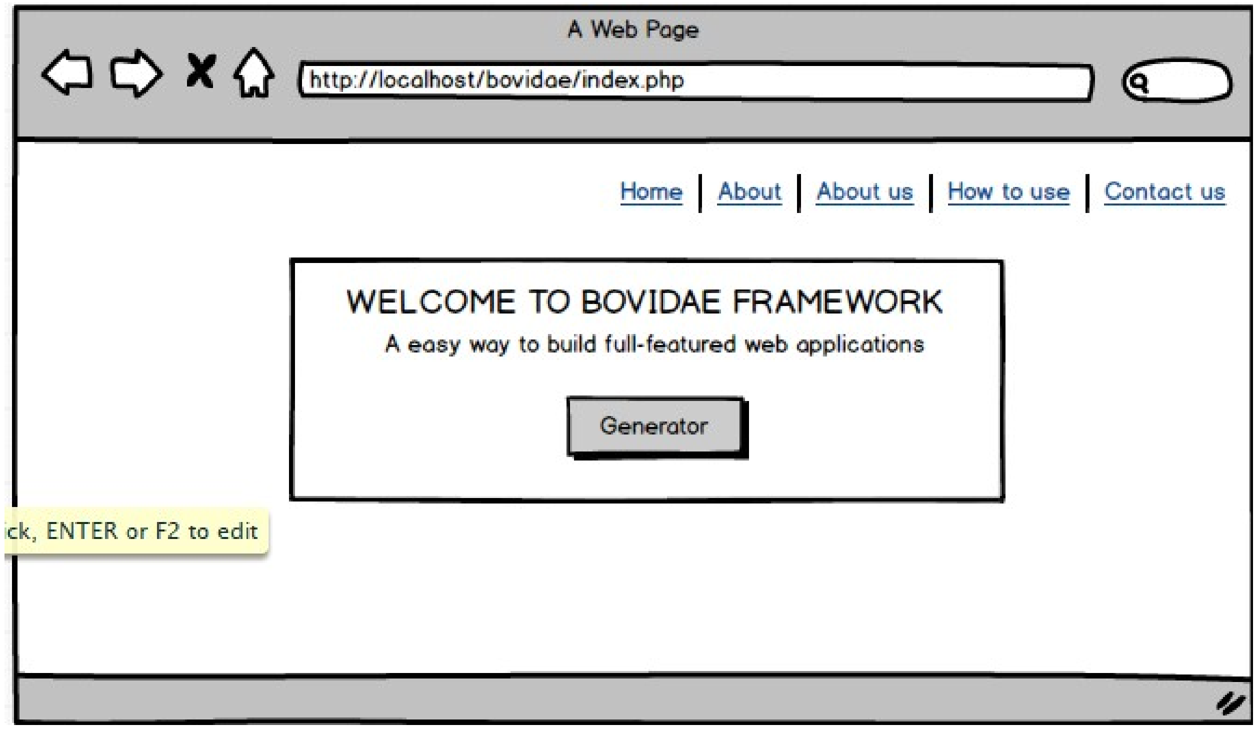
1. The developer is able to see the interface of view.

Appendix D: Wireframing

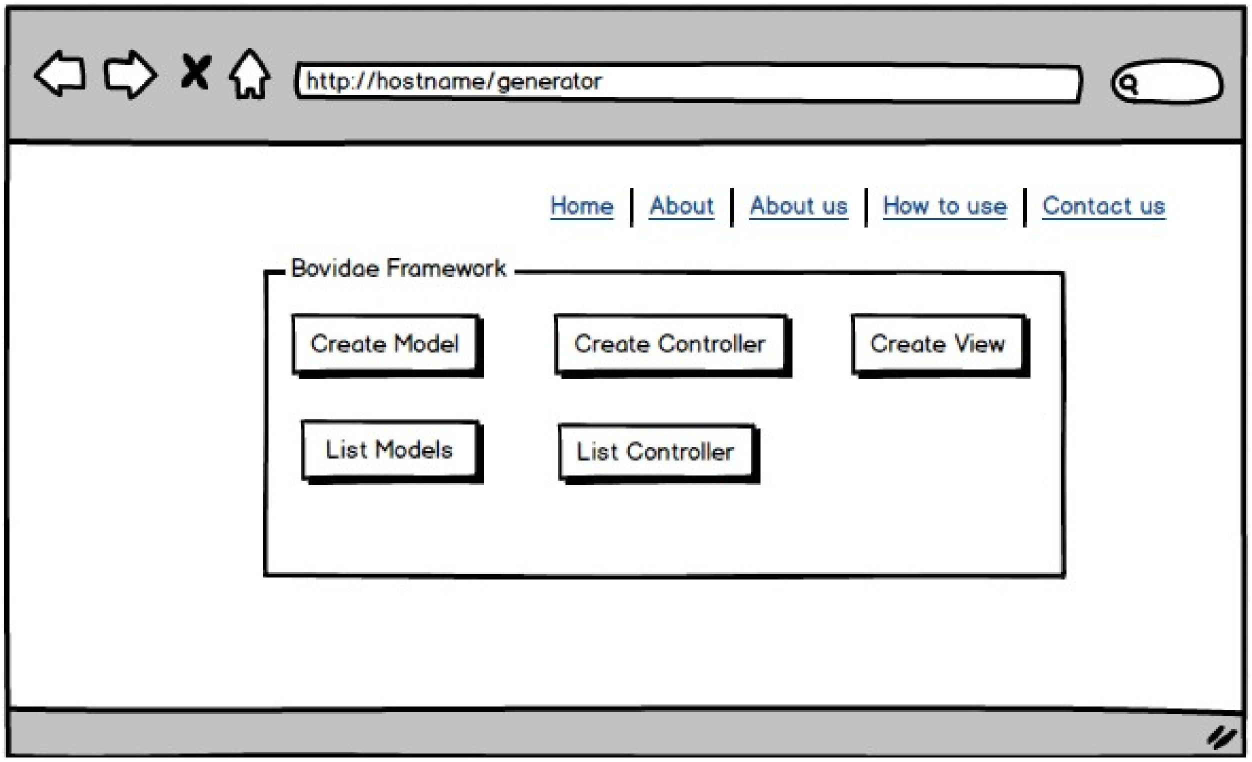
5.6.1 Main Home Page


A Web Page
http://www.bovidae.com
Home | Products | Company | Blog
WELCOME TO BOVIDAE FRAMEWORK
A easy way to build full-featured web applications
Download

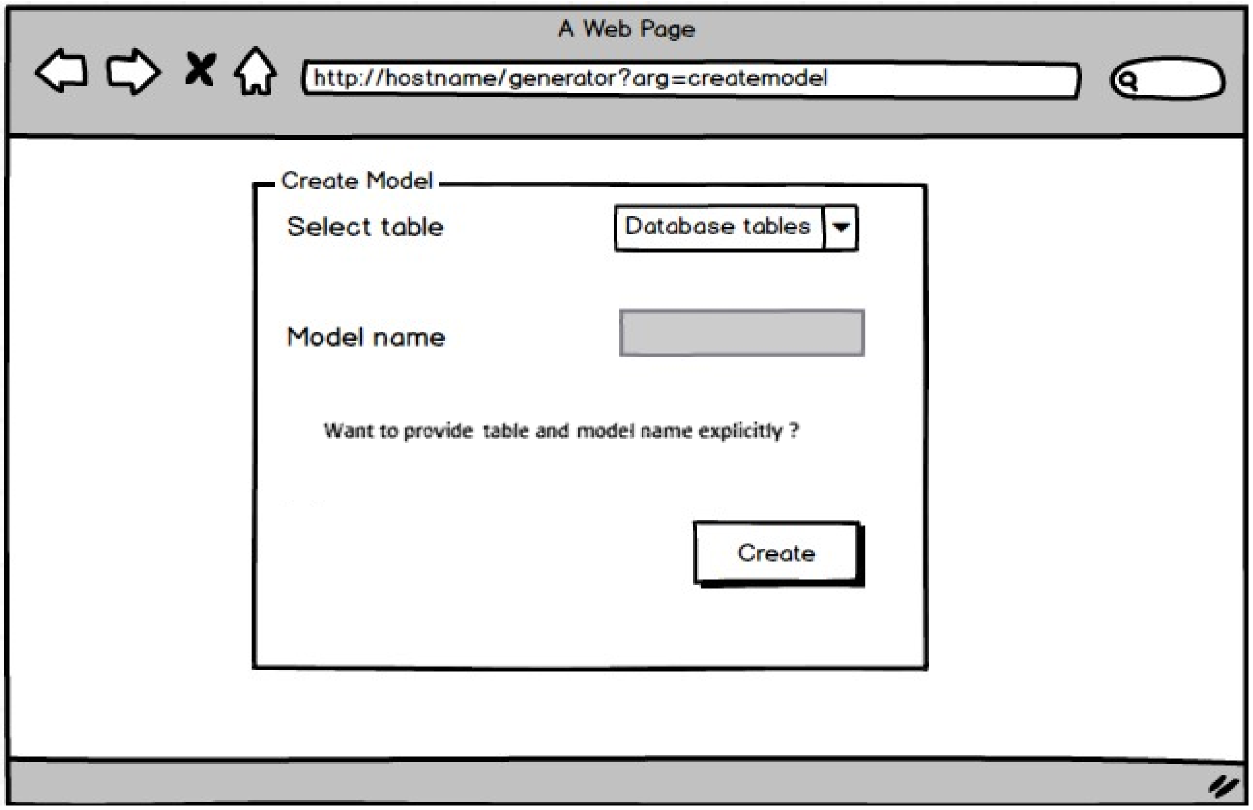
5.6.2 Development Page


A Web Page
http://localhost/bovidae/index.php
Home | About | About us | How to use | Contact us
WELCOME TO BOVIDAE FRAMEWORK
A easy way to build full-featured web applications
Generator
ick, ENTER or F2 to edit

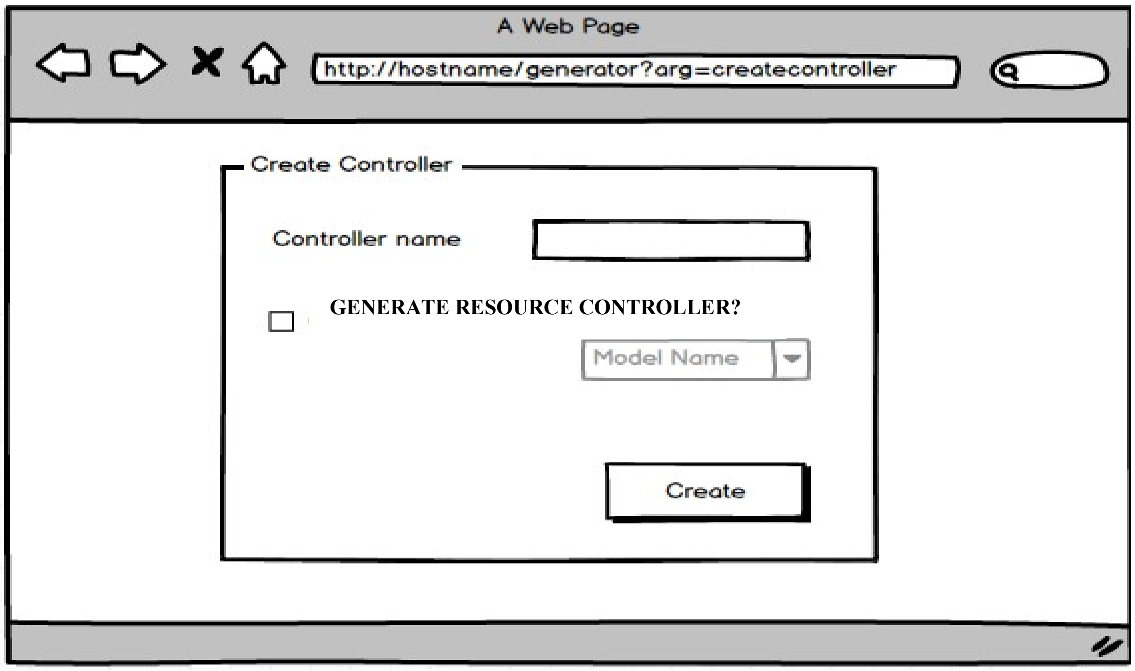
5.6.3 Generator



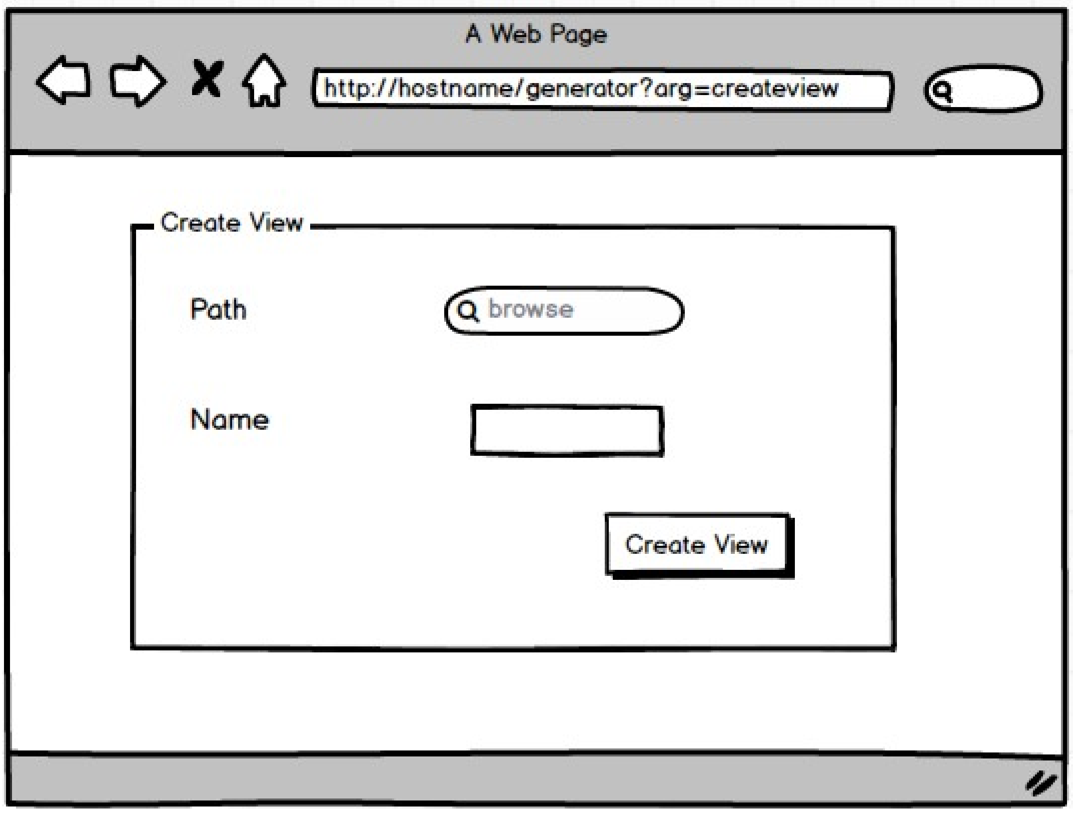
5.6.4 Create Model


A Web Page
http://hostname/generator?arg=createmodel
Create Model
Select table
Database tables
Model name
Want to provide table and model name explicitly ?
Create

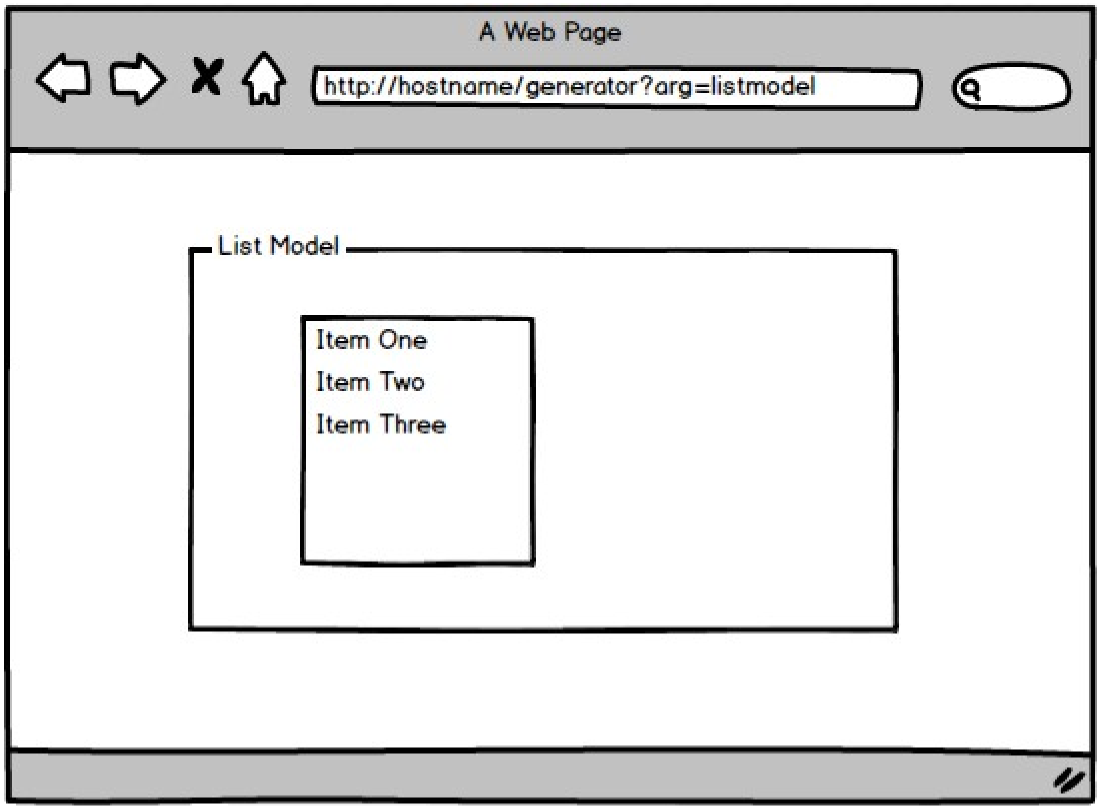
5.6.5 Create Controller


A Web Page
http://hostname/generator?arg=createcontroller
Create Controller
Controller name
GENERATE RESOURCE CONTROLLER?
Model Name
Create

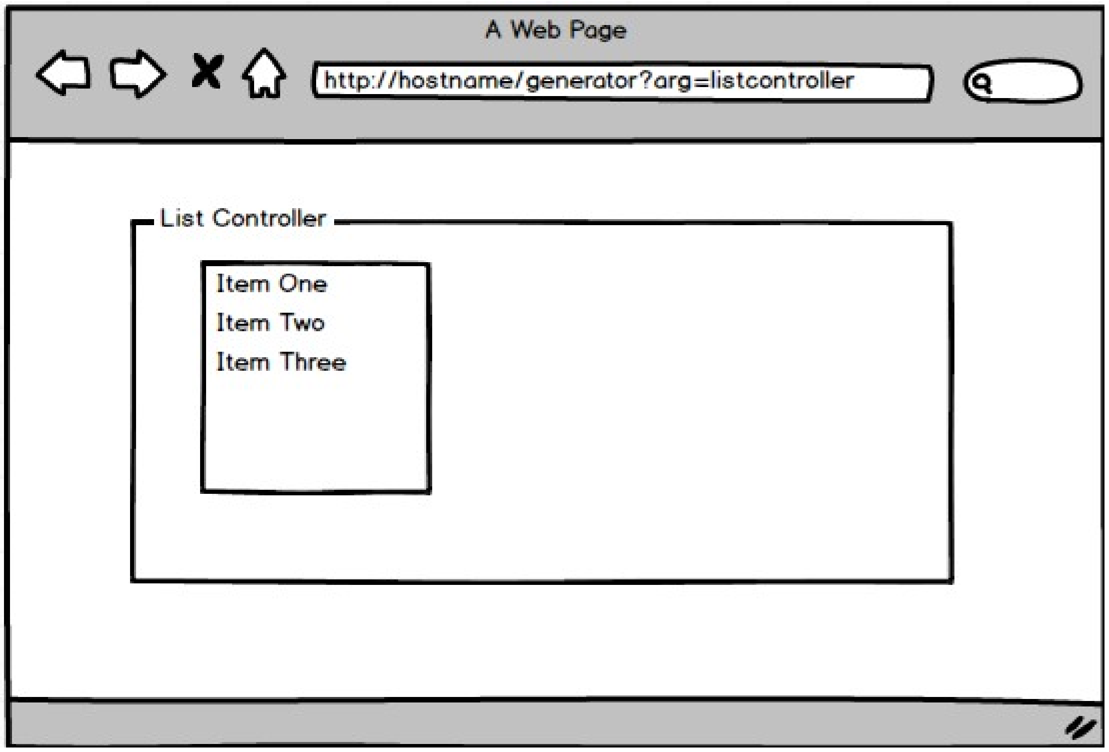
5.6.5 Create View



5.6.6 List All Models



5.6.7 List All Controller



Appendix E:

Roles & Responsibility Matrix:

The purpose of roles & responsibility matrix is to identify who will do what.

Table 2: Roles & Responsibility Matrix

WBS #	**WBS Deliverable**	**Activity #**	**Activity to Complete the Deliverable**	**Duration (# of Days)**	**Responsible Team Member(s) & Role(s)**
1	**Project Management**	1.1	Work breakdown structure (WBS)	3	Kainat Khalid, Asma Malik, Amna shehzadee
		1.2	Roles and responsibility Matrix	2	Kainat Khalid, Asma Malik, Amna shehzadee
		1.3	Change Control System	1	Kainat Khalid, Asma Malik, Amna shehzadee
2	**Reports/Documentation**	2.1	Final Documentation Introduction	1	Kainat Khalid, Asma Malik, Amna shehzadee
		2.2	Literature / Markey Survey 2.2.1 Questionnaires 2.2.2 Find Interviewee 2.2.3 Conduct Interview	2	Kainat Khalid, Asma Malik, Amna shehzadee
		2.3	Requirements Analysis	5	Kainat Khalid,

			2.3.1 Eliciting requirement 2.3.2 Analyzing requirements 2.3.3 Document the requirements(SRS) 2.3.4 Recording requirements 2.3.5 Use case diagram 2.3.6 ER diagram 2.3.7		Asma Malik, Amna shehzadee
		2.4	System Design 2.4.1 ER diagram 2.4.2 Use case diagram 2.4.3 Domain Model 2.4.4 Architecture Diagram 2.4.5 Deployment Diagram	5	Kainat Khalid, Asma Malik, Amna shehzadee
		2.5	Implementation 2.4.1 Prototype	5	Kainat Khalid, Asma Malik, Amna shehzadee
		2.6	Testing & Performance Evaluation 2.6.1 Black box	17	Kainat Khalid, Asma Malik, Amna shehzadee

			testing 2.6.2 Check all the links 2.6.3 Functional testing 2.6.4 Validate your HTML/CSS 2.6.5 Database check 2.6.6 Performance Testing		
		2.7	Conclusion & Outlook	2	Kainat Khalid, Asma Malik, Amna shehzadee
		2.8	End User Documentation	3	Kainat Khalid, Asma Malik, Amna shehzadee
		2.10	System Administrator Documentation	2	Kainat Khalid, Asma Malik, Amna shehzadee
3	**System**	3.1	Development Environment 3.1.1 IDE 3.1.2 Version Control 3.1.3 Server 3.1.4 Database	25	Kainat Khalid, Asma Malik, Amna shehzadee
		3.2	Presentation Layer 3.2.1 Developer	15	Kainat Khalid, Asma Malik, Amna

					shehzadee
		3.3.	Business Logic Layer 3.3.1 Implementation	12	Kainat Khalid, Asma Malik, Amna shehzadee
		3.4	Data Management Layer	10	Kainat Khalid, Asma Malik, Amna shehzadee
		3.5	Physical Layer 3.5.1 Database	15	Kainat Khalid, Asma Malik, Amna shehzadee

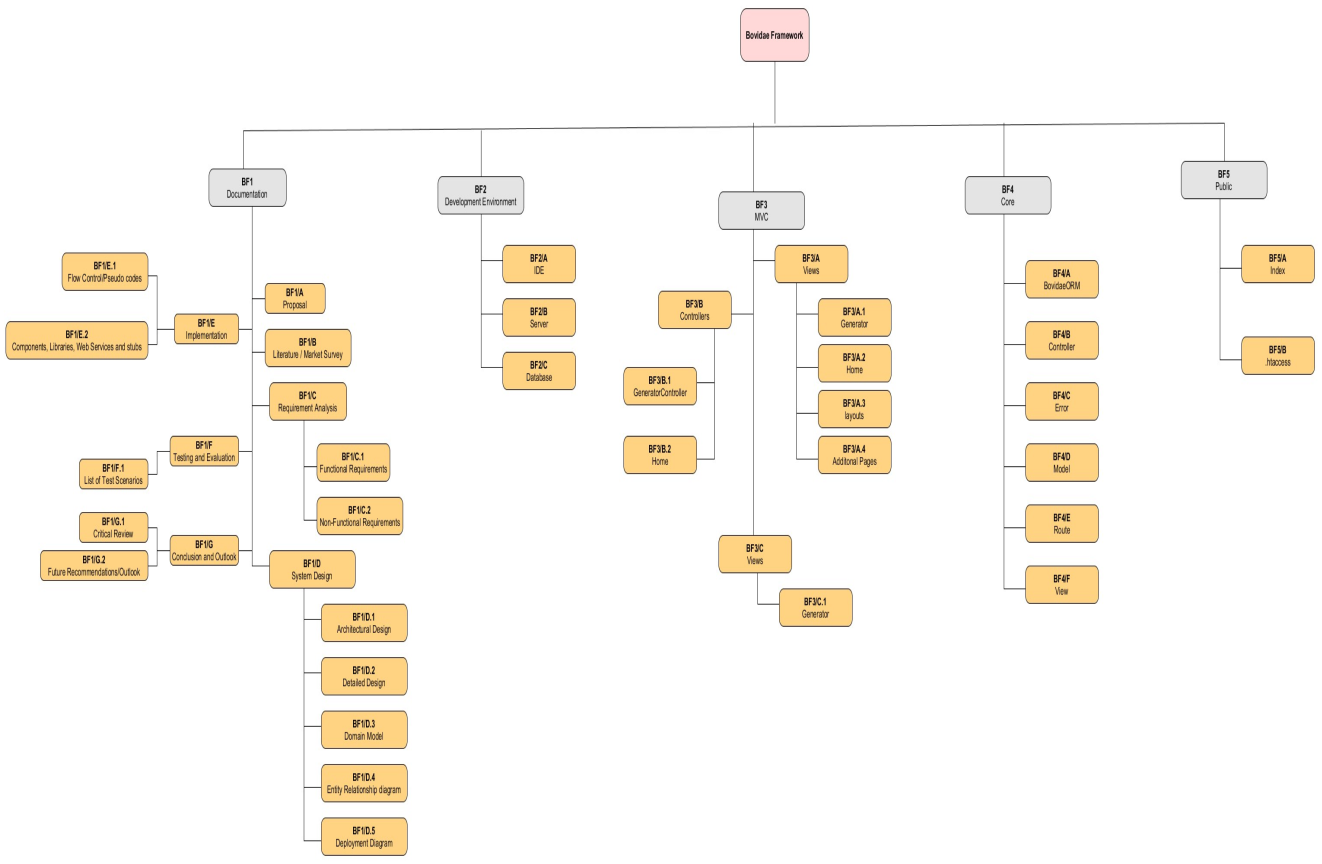
Appendix F:

Work Breakdown Structure of Bovidae Framework

Table 3: Work Breakdown Structure (WBS)

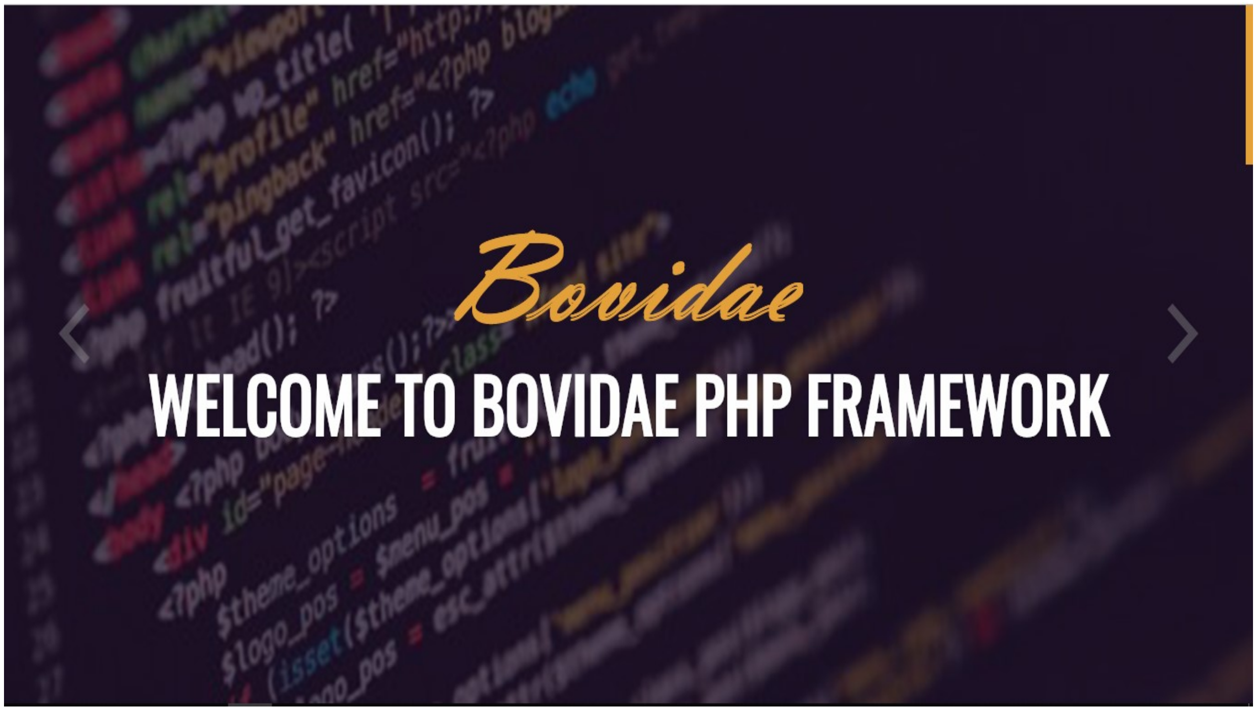
Task Name	Duration	Start	Finish	WBS
Documentation	**45 days**	**Wed 14/03/18**	**Tue 15/05/18**	**BF1**
Proposal	3 days	Wed 14/03/18	Fri 16/03/18	BF1/A
Literature / Market Survey	6 days	Fri 16/03/18	Fri 23/03/18	BF1/B
Requirement Analysis	**9 days**	**Sat 17/03/18**	**Wed 28/03/18**	**BF1/C**
Functional Requirements	7 days	Mon 19/03/18	Tue 27/03/18	BF1/C.1
Non Functional Requirements	2 days	Tue 27/03/18	Wed 28/03/18	BF1/C.2
System Design	**8 days**	**Thu 29/03/18**	**Mon 09/04/18**	**BF1/D**
Architectural Design	2 days	Thu 29/03/18	Fri 30/03/18	BF1/D.1
Detailed Design	1 day	Fri 30/03/18	Fri 30/03/18	BF1/D.2
Domain Model	2 days	Tue 03/04/18	Wed 04/04/18	BF1/D.4
Deployment Diagram	2 days	Fri 06/04/18	Mon 09/04/18	BF1/D.5
Entity Relationship diagram	2 days	Mon 02/04/18	Tue 03/04/18	BF1/D.3
Implementation	**10 days**	**Thu 09/08/18**	**Wed 22/08/18**	**BF1/E**
Flow Control/Pseudo codes	4 days	Thu 09/08/18	Tue 14/08/18	BF1/E.1
Components, Libraries, Web Services and stubs	3 days	Mon 20/08/18	Wed 22/08/12	BF1/E.3
Deployment Environment	4 days	Thu 09/08/18	Tue 14/08/18	BF1/E.2
Testing and Evaluation	**10 days**	**Mon 12/11/18**	**Fri 23/11/18**	**BF1/F**
List of Test Scenarios	10 days	Mon 12/11/18	Fri 23/11/18	BF1/F.1
Conclusion and Outlook	**2 days**	**Thu 29/11/18**	**Fri 30/11/18**	**BF1/G**
Critical Review	2 days	Thu 29/11/18	Fri 30/11/18	BF1/G.1
Future Recommendations/Outlook	2 days	Thu 29/11/18	Fri 30/11/18	BF1/G.2
Development Environment	**5 days**	**Thu 09/08/18**	**Wed 15/08/18**	**BF2**
IDE	3 days	Thu 09/08/18	Mon 13/08/18	BF2/A
Server	2 days	Thu 09/08/18	Fri 10/08/18	BF2/B
Database	5 days	Thu 09/08/18	Wed 15/08/18	BF2/C
MVC	**50 days**	**Thu 09/08/18**	**Wed 17/10/18**	**BF3**
Views	**20 days**	**Fri 10/08/18**	**Thu 06/09/18**	**BF3/A**
Generator	12 days	Fri 10/08/18	Mon 27/08/18	BF3/A.1
Home	4 days	Mon 27/08/18	Thu 30/08/18	BF3/A.2
Layouts	2 days	Fri 24/08/18	Mon 27/08/18	BF3/A.3
Additonal Pages	2 days	Thu 05/09/18	Fri 05/0918	BF3/A.4
Controllers	**20 days**	**Thu 06/09/18**	**Wed 26/09/18**	**BF3/B**

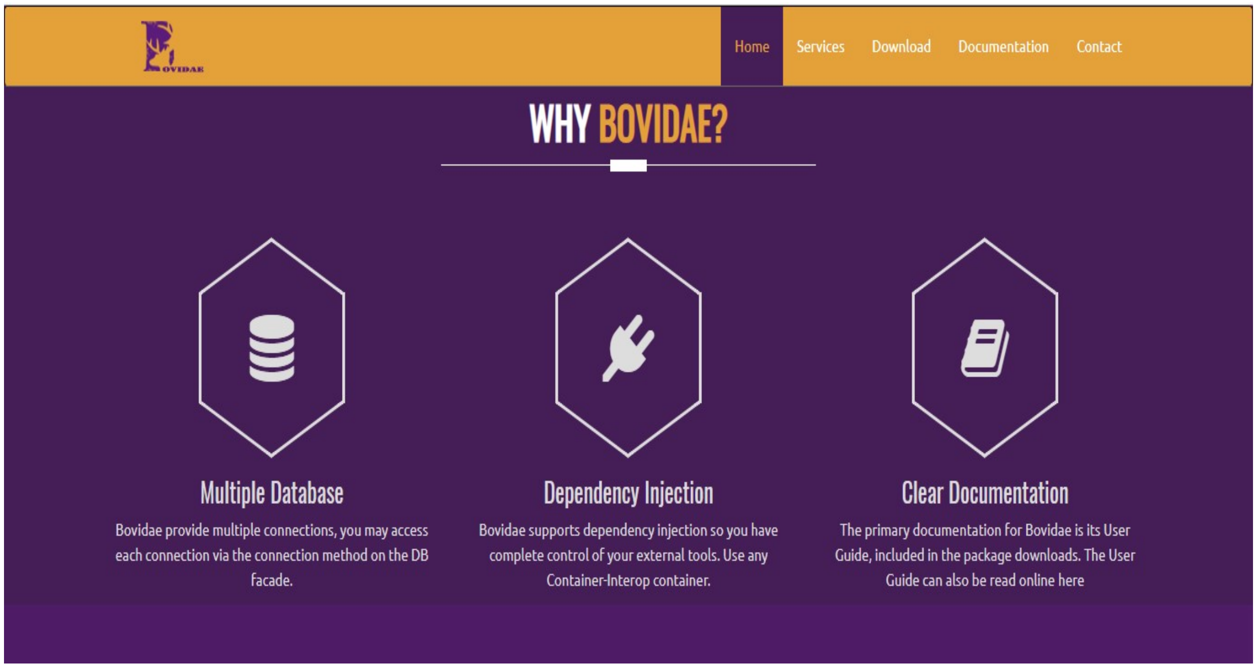
GeneratorController	10 days	Mon 06/09/18	Fri 21/09/18	BF3/B.1
Home	2 days	Thu 24/09/18	Fri 26/09/18	BF3/B.2
Model	**10 days**	**Thu 27/09/18**	**Wed 17/10/18**	**BF3/C**
Generator	10 days	Thu 27/09/18	Wed 17/10/18	BF3/C.1
Core	**30 days**	**Wed 19/09/18**	**Tue 30/10/18**	**BF4**
BovidaeORM	7 days	Wed 19/09/18	Thu 27/09/18	BF4/A
Controller	5 days	Thu 27/09/18	Wed 03/10/18	BF4/B
Error	3 days	Thu 04/10/18	Mon 08/10/18	BF4/C
Model	5 days	Tue 09/10/18	Mon 15/10/18	BF4/D
Route	10 days	Mon 15/10/18	Fri 26/10/18	BF4/E
View	5 days	Thu 25/10/18	Wed 31/10/18	BF4/F
Public	**15 days**	**Fri 23/11/18**	**Thu 13/12/18**	**BF5**
Index	10 days	Fri 23/11/18	Thu 06/12/18	BF5/A
.htaccess	6 days	Thu 06/12/18	Thu 13/12/18	BF5/B


Bovidae Framework
BF1 Documentation
BF1/A Proposal
BF1/B Literature / Market Survey
BF1/C Requirement Analysis
BF1/C.1 Functional Requirements
BF1/C.2 Non-Functional Requirements
BF1/D System Design
BF1/D.1 Architectural Design
BF1/D.2 Detailed Design
BF1/D.3 Domain Model
BF1/D.4 Entity Relationship diagram
BF1/D.5 Deployment Diagram
BF1/E Implementation
BF1/E.1 Flow Control/Pseudo codes
BF1/E.2 Components, Libraries, Web Services and stubs
BF1/F Testing and Evaluation
BF1/F.1 List of Test Scenarios
BF1/G Conclusion and Outlook
BF1/G.1 Critical Review
BF1/G.2 Future Recommendations/Outlook
BF2 Development Environment
BF2/A IDE
BF2/B Server
BF2/C Database
BF3 MVC
BF3/A Views
BF3/A.1 Generator
BF3/A.2 Home
BF3/A.3 layouts
BF3/A.4 Additonal Pages
BF3/B Controllers
BF3/B.1 GeneratorController
BF3/B.2 Home
BF3/C Views
BF3/C.1 Generator
BF4 Core
BF4/A BovidaeORM
BF4/B Controller
BF4/C Error
BF4/D Model
BF4/E Route
BF4/F View
BF5 Public
BF5/A Index
BF5/B .htaccess

Appendix G:

Website Prototype





WHY BOVIDAE?

Multiple Database

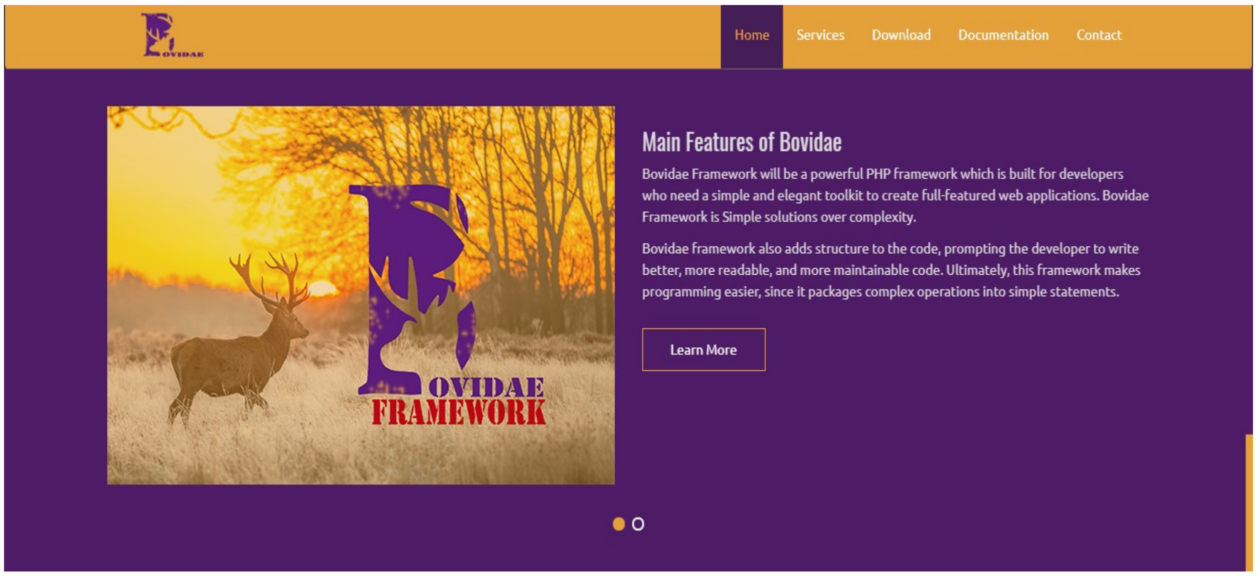
Bovidae provide multiple connections, you may access each connection via the connection method on the DB facade.

Dependency Injection

Bovidae supports dependency injection so you have complete control of your external tools. Use any Container-Interop container.

Clear Documentation

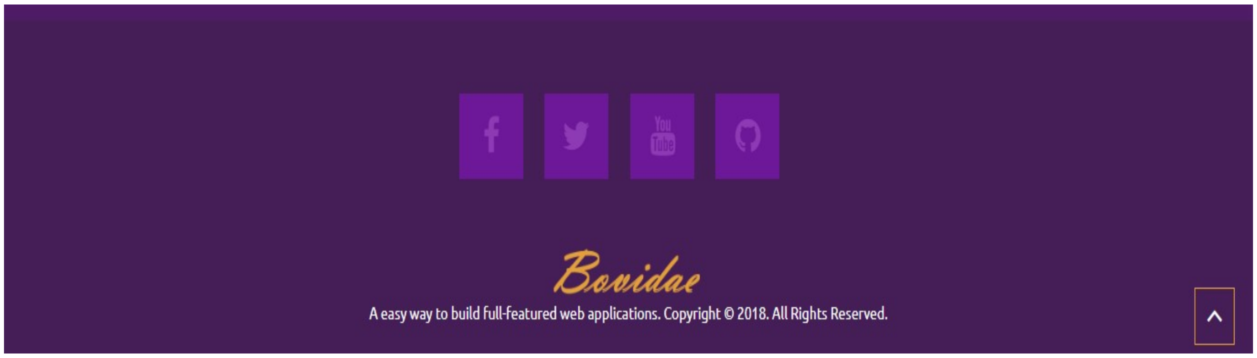
The primary documentation for Bovidae is its User Guide, included in the package downloads. The User Guide can also be read online here



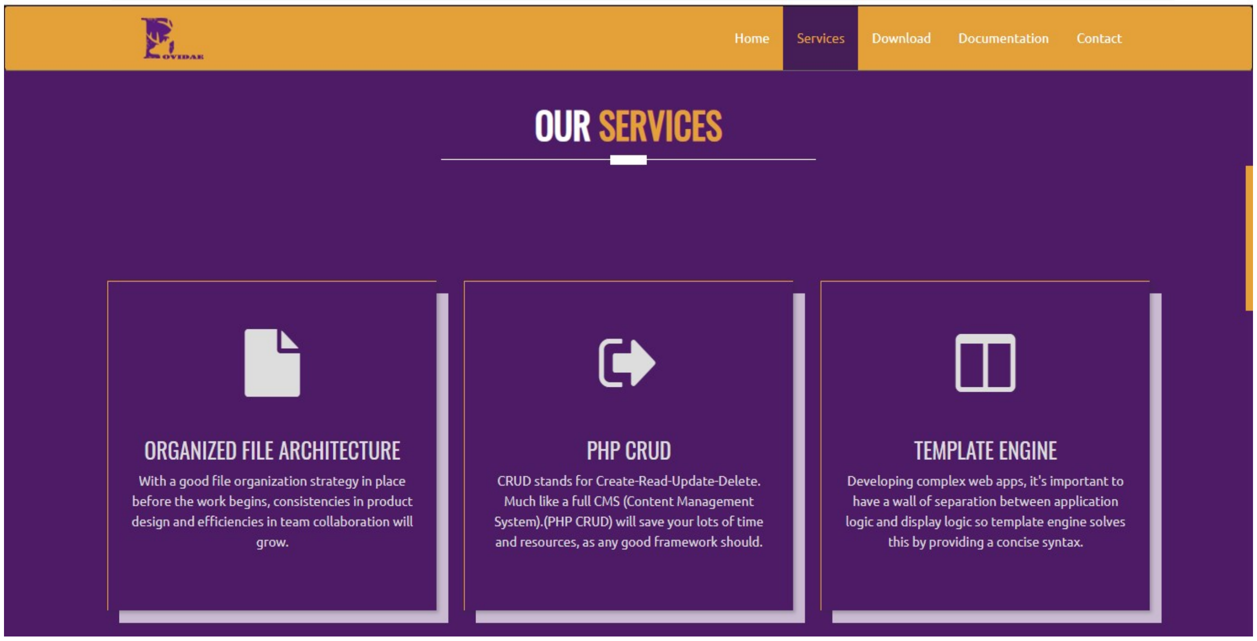
Main Features of Bovidae

Bovidae Framework will be a powerful PHP framework which is built for developers who need a simple and elegant toolkit to create full-featured web applications. Bovidae Framework is Simple solutions over complexity.

Bovidae framework also adds structure to the code, prompting the developer to write better, more readable, and more maintainable code. Ultimately, this framework makes programming easier, since it packages complex operations into simple statements.



A easy way to build full-featured web applications. Copyright © 2018. All Rights Reserved.



OUR SERVICES

ORGANIZED FILE ARCHITECTURE

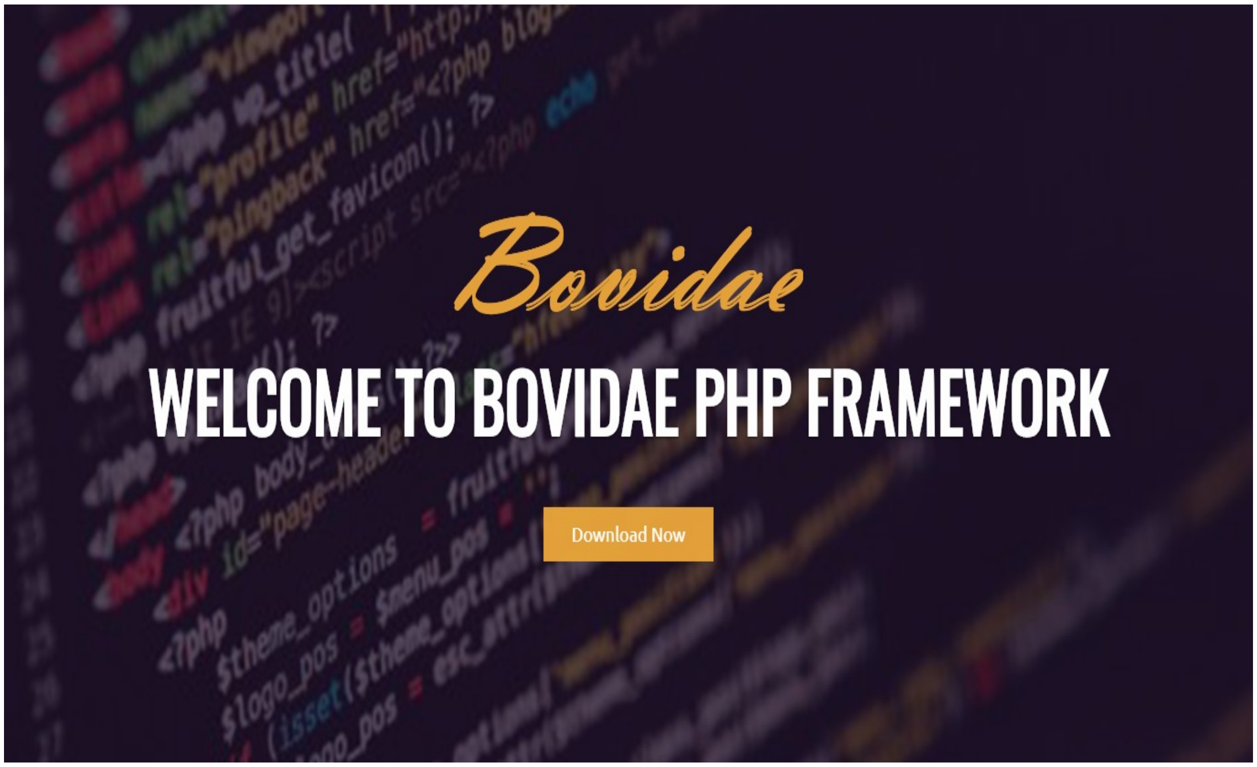
With a good file organization strategy in place before the work begins, consistencies in product design and efficiencies in team collaboration will grow.

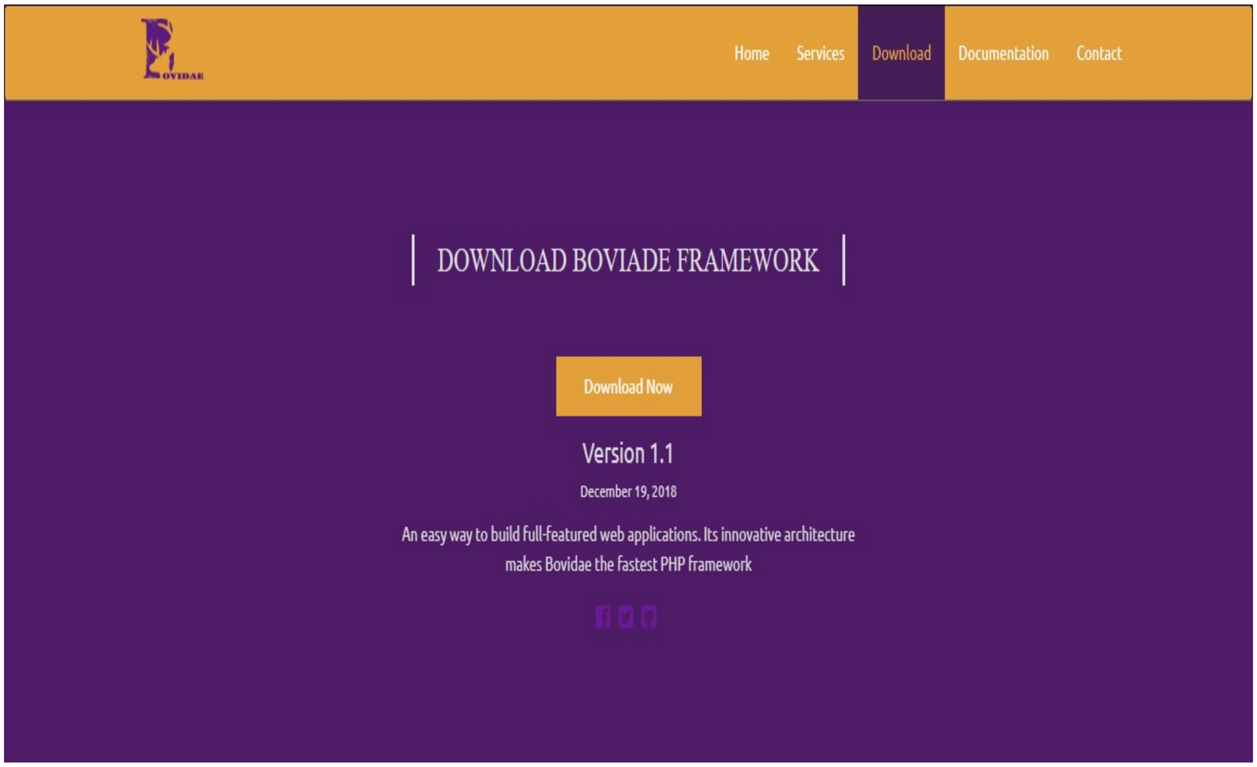
PHP CRUD

CRUD stands for Create-Read-Update-Delete. Much like a full CMS (Content Management System).(PHP CRUD) will save your lots of time and resources, as any good framework should.

TEMPLATE ENGINE

Developing complex web apps, it's important to have a wall of separation between application logic and display logic so template engine solves this by providing a concise syntax.



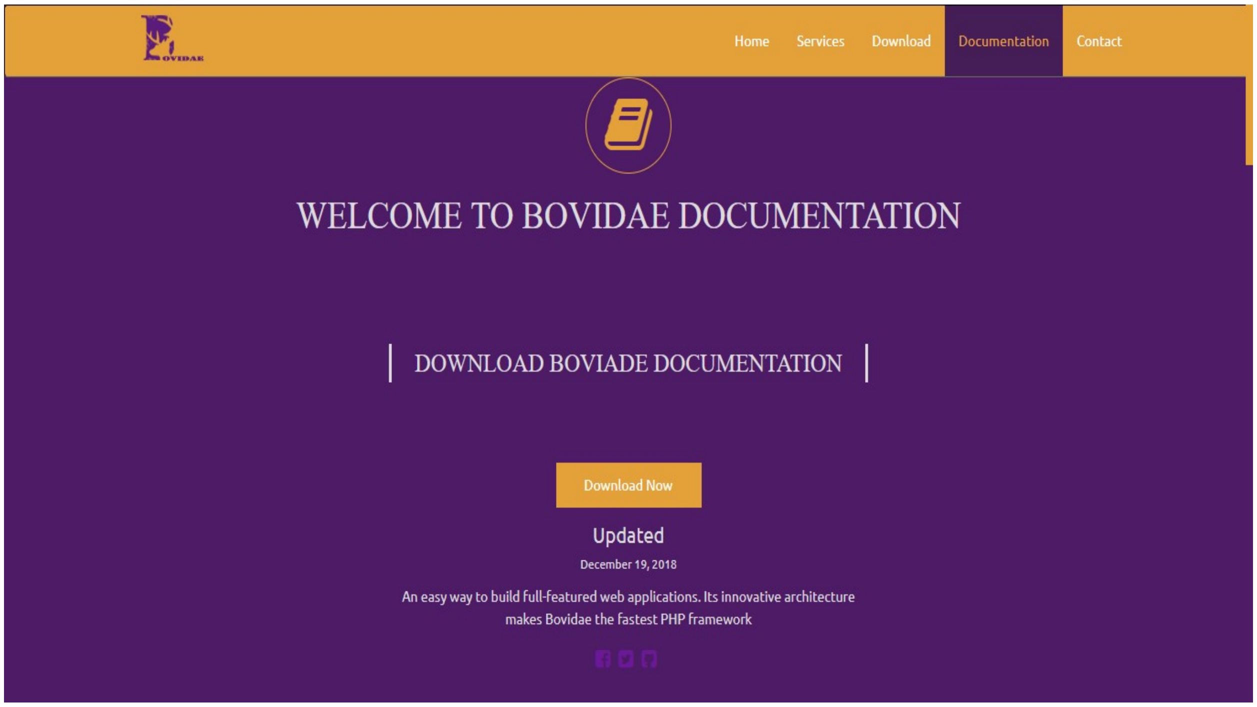


DOWNLOAD BOVIADE FRAMEWORK

Version 1.1

December 19, 2018

An easy way to build full-featured web applications. Its innovative architecture makes Bovidae the fastest PHP framework



Home Services Download Documentation Contact

WELCOME TO BOVIDAE DOCUMENTATION

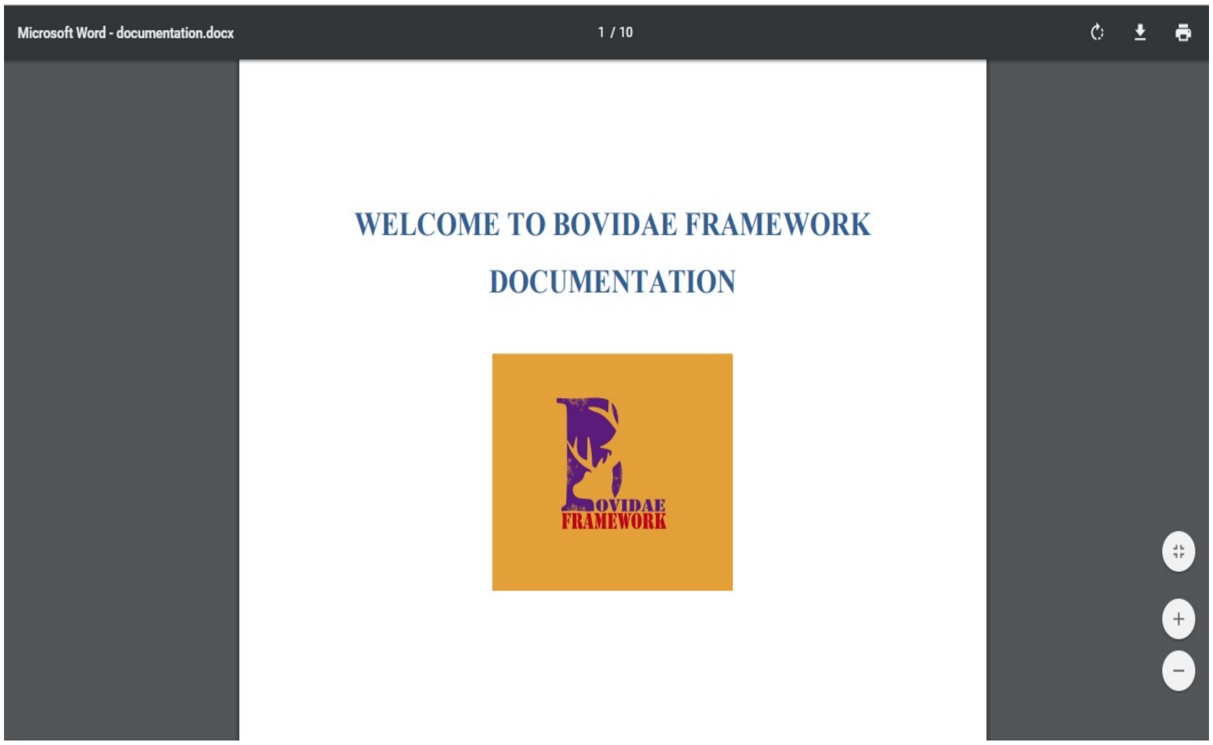
DOWNLOAD BOVIADE DOCUMENTATION

Download Now

Updated

December 19, 2018

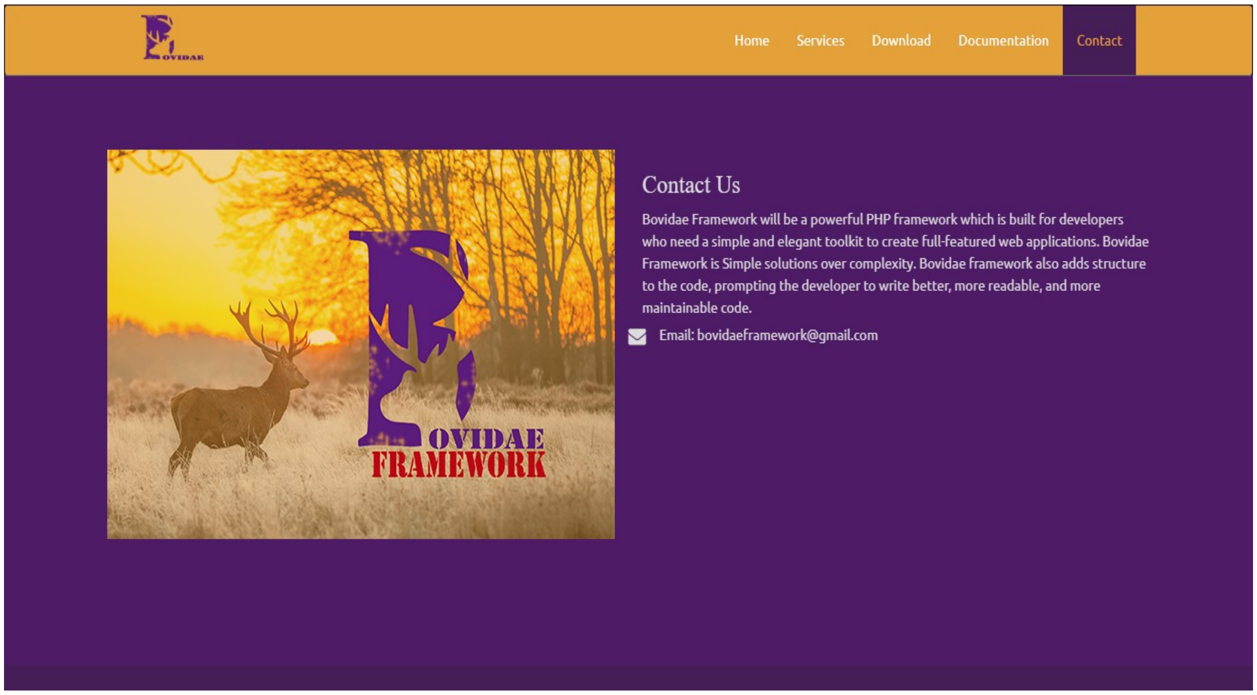
An easy way to build full-featured web applications. Its innovative architecture makes Bovidae the fastest PHP framework



Microsoft Word - documentation.docx

1 / 10

WELCOME TO BOVIDAE FRAMEWORK DOCUMENTATION

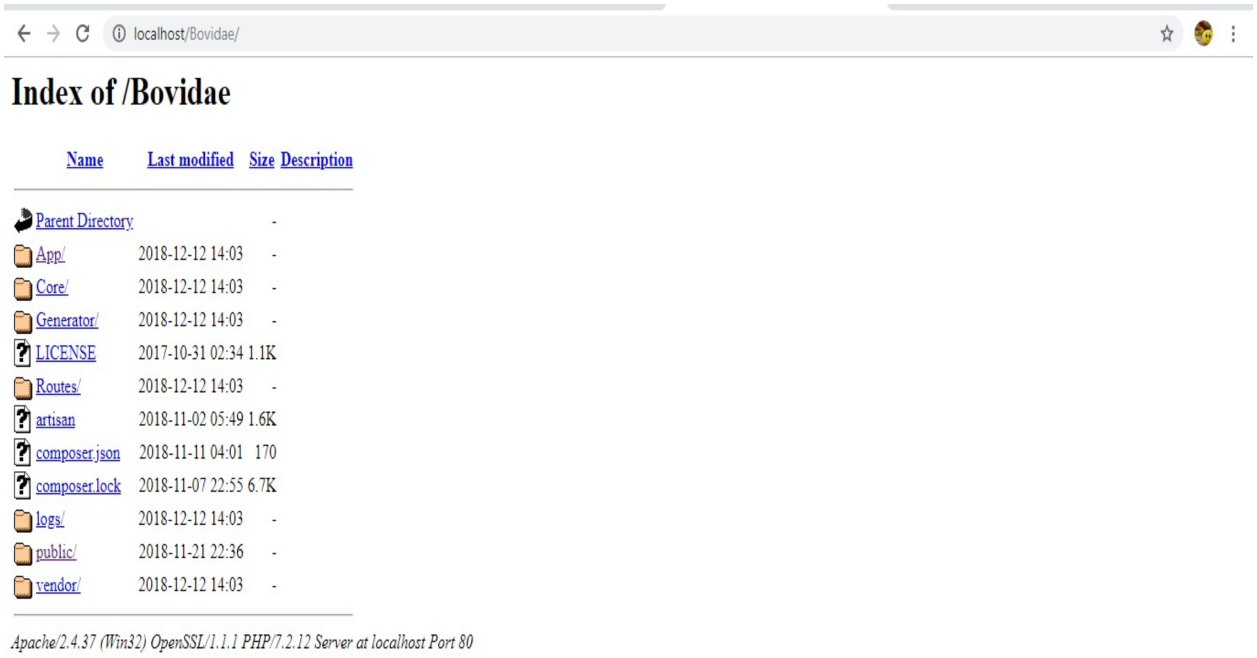


Contact Us

Bovidae Framework will be a powerful PHP framework which is built for developers who need a simple and elegant toolkit to create full-featured web applications. Bovidae Framework is Simple solutions over complexity. Bovidae framework also adds structure to the code, prompting the developer to write better, more readable, and more maintainable code.

Email: bovidaeframework@gmail.com

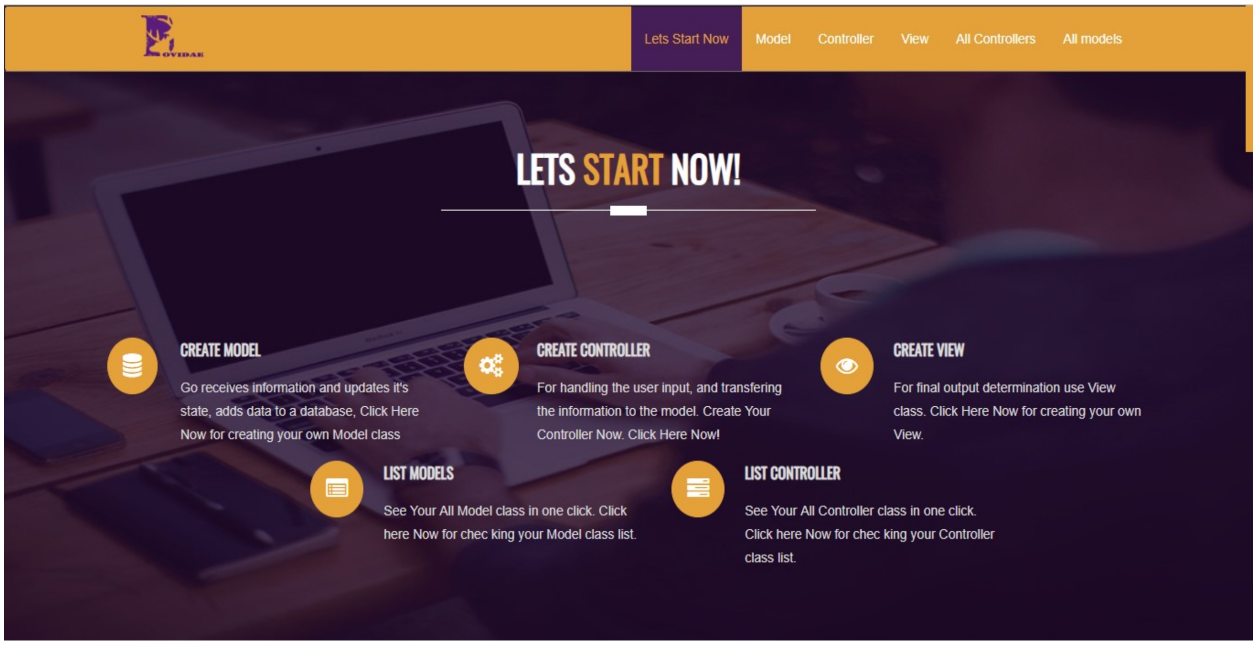
Framework Views:



Index of /Bovidae

```
     Name          Last modified     Size Description
  Parent Directory                     -
  App/             2018-12-12 14:03    -
  Core/            2018-12-12 14:03    -
  Generator/       2018-12-12 14:03    -
  LICENSE          2017-10-31 02:34  1.1K
  Routes/          2018-12-12 14:03    -
  artisan          2018-11-02 05:49  1.6K
  composer.json    2018-11-11 04:01   170
  composer.lock    2018-11-07 22:55  6.7K
  logs/            2018-12-12 14:03    -
  public/          2018-11-21 22:36    -
  vendor/          2018-12-12 14:03    -
```

Apache/2.4.37 (Win32) OpenSSL/1.1.1 PHP/7.2.12 Server at localhost Port 80



CREATE MODEL

Go receives information and updates it's state, adds data to a database, Click Here Now for creating your own Model class

CREATE CONTROLLER

For handling the user input, and transfering the information to the model. Create Your Controller Now. Click Here Now!

CREATE VIEW

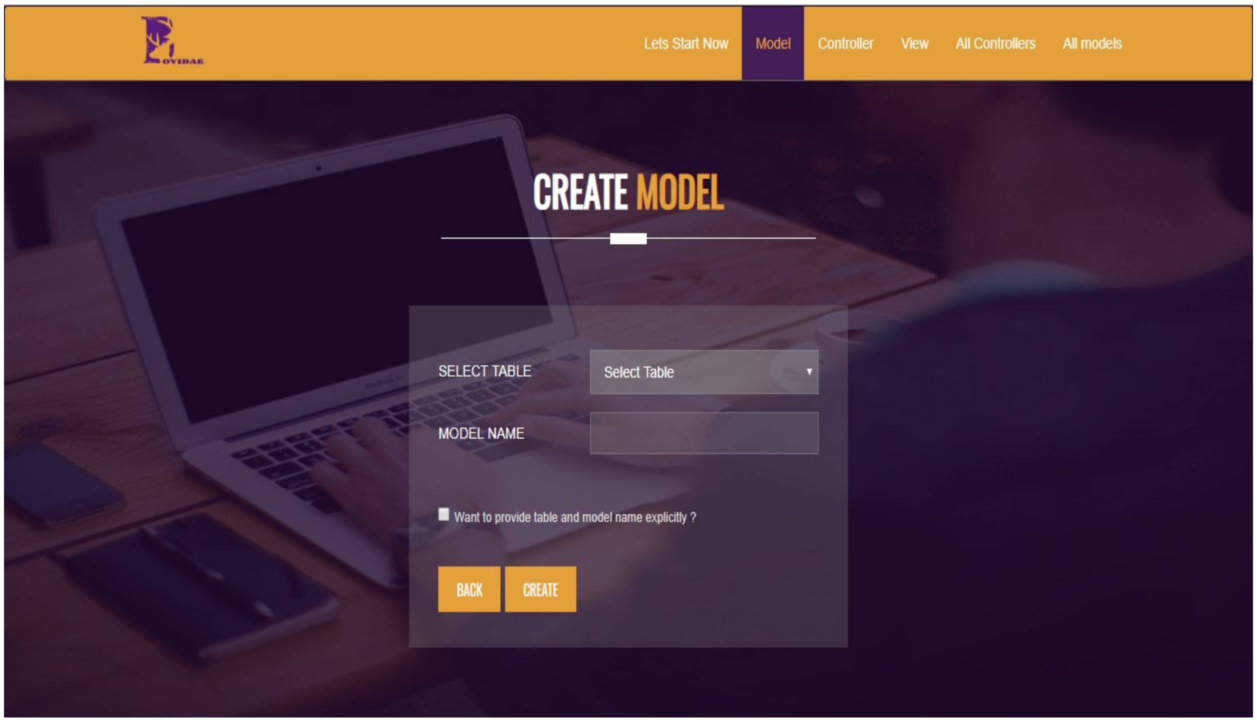
For final output determination use View class. Click Here Now for creating your own View.

LIST MODELS

See Your All Model class in one click. Click here Now for chec king your Model class list.

LIST CONTROLLER

See Your All Controller class in one click. Click here Now for chec king your Controller class list.



SELECT TABLE	Select Table
MODEL NAME	

Want to provide table and model name explicitly ?

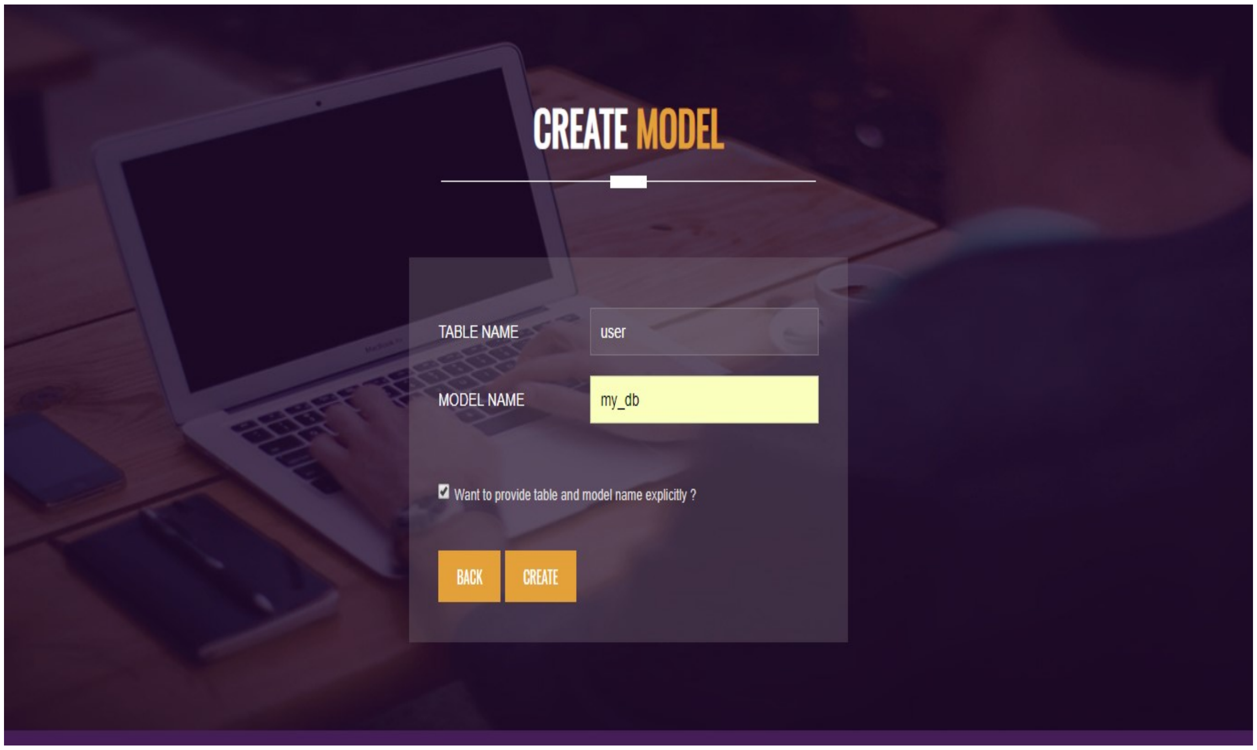
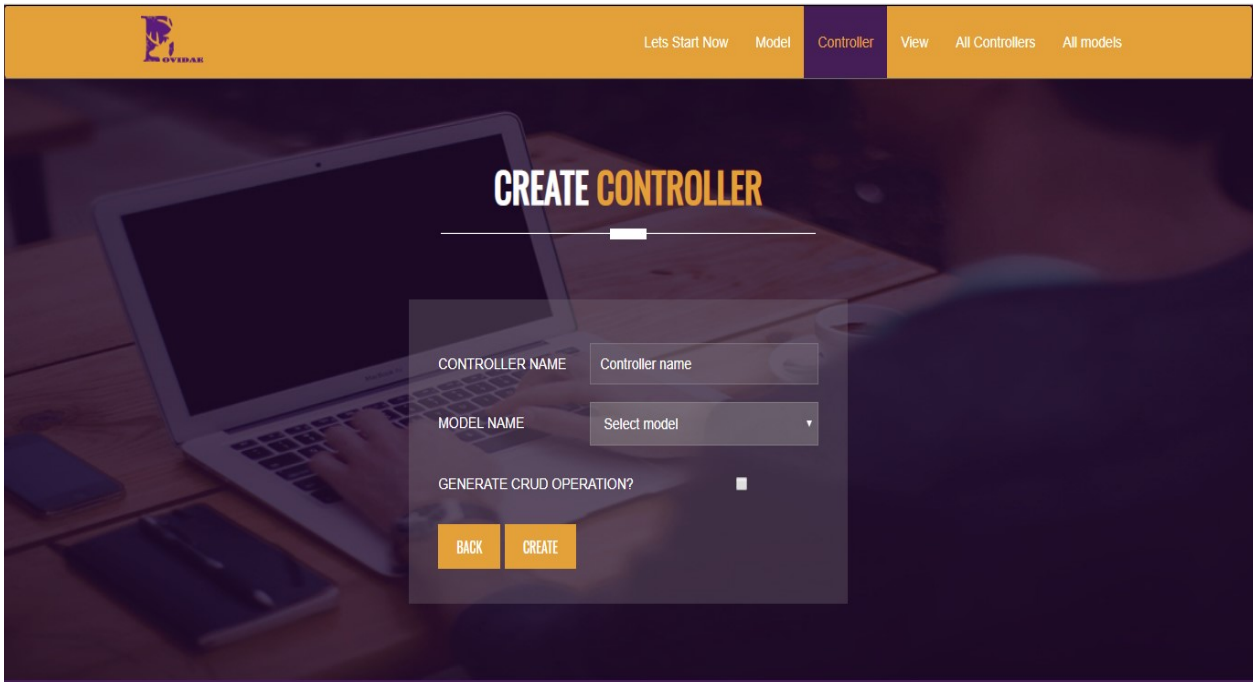

CREATE MODEL

TABLE NAME user

MODEL NAME my_db

☑ Want to provide table and model name explicitly ?

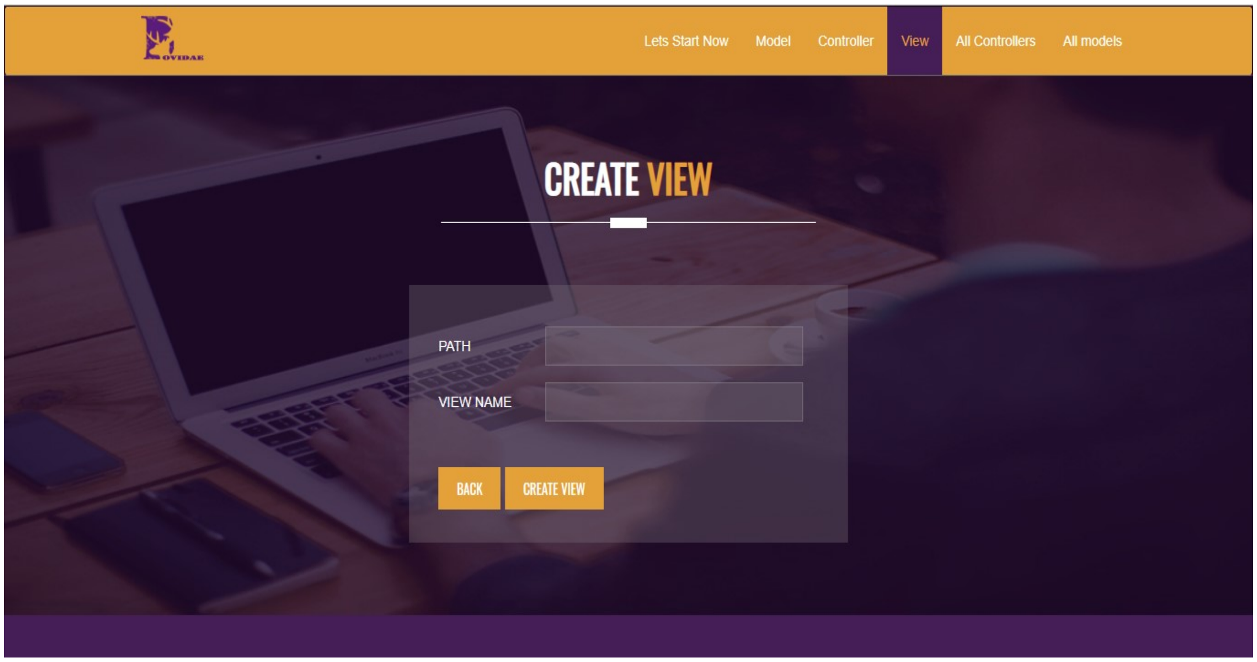


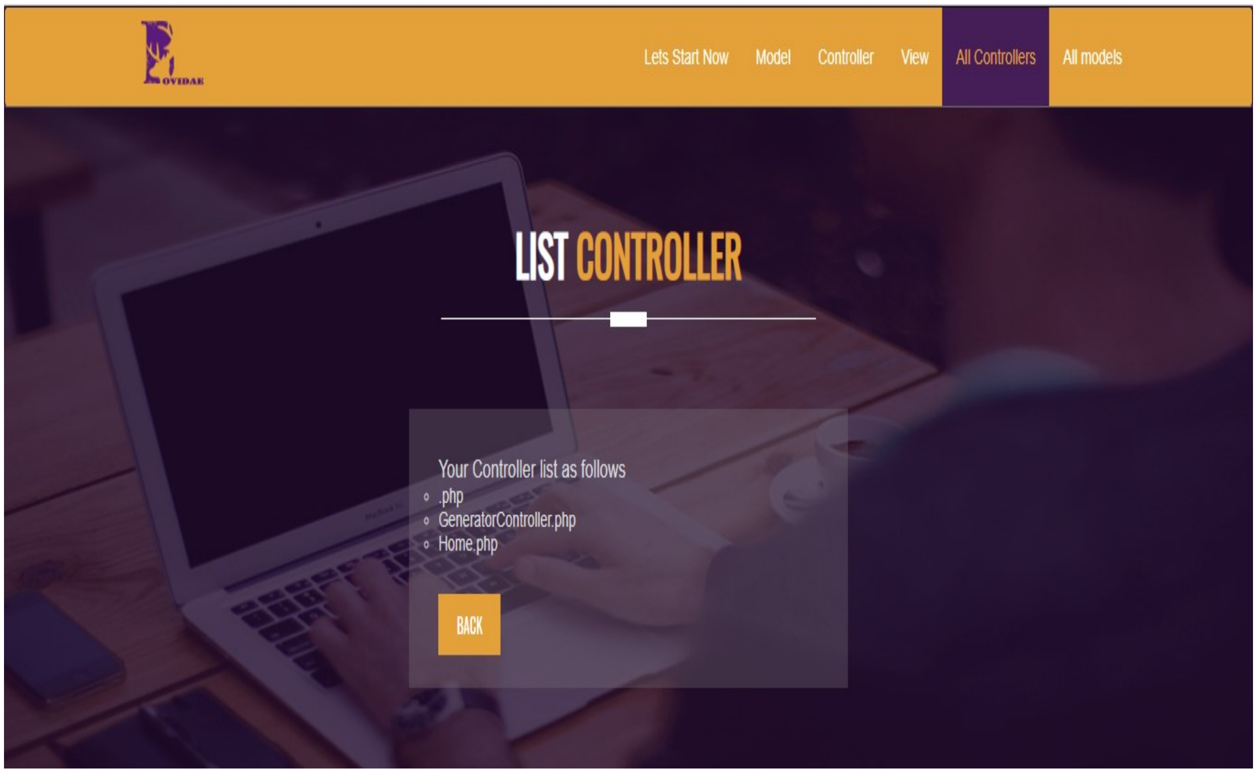
CREATE CONTROLLER

CONTROLLER NAME Controller name

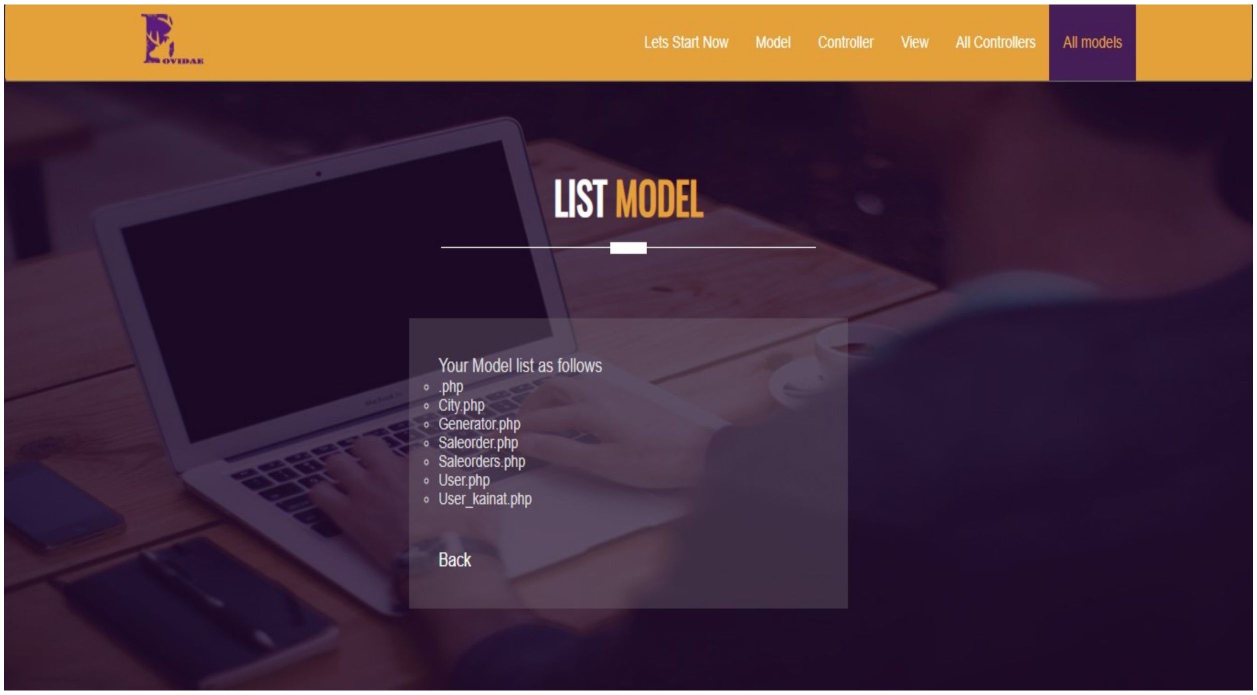
MODEL NAME Select model

GENERATE CRUD OPERATION? ☐





Your Controller list as follows
- .php
- GeneratorController.php
- Home.php

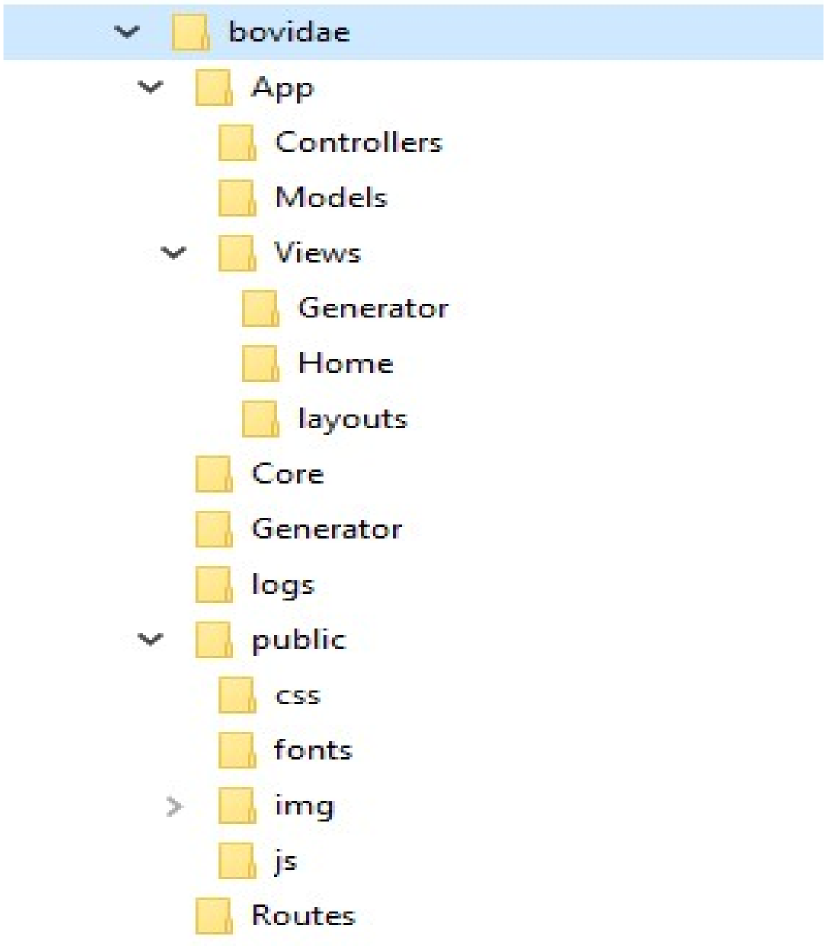


Your Model list as follows

- .php
- City.php
- Generator.php
- Saleorder.php
- Saleorders.php
- User.php
- User_kainat.php

Back

Bovidae Framework Directory:



```
bovidae
  App
    Controllers
    Models
    Views
      Generator
      Home
      layouts
  Core
  Generator
  logs
  public
    css
    fonts
    img
    js
  Routes
```