**Project Report**

**On**

**CROP MANAGEMENT SYSTEM**

****

**SUBMITTED BY**

**Miss. P. Geeta Sri Regd. No. 314175710175**

**Miss. P. Kalyani Regd. No. 314175710176**

**Mr. P. Raj Kiran Regd. No. 314175710165**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**S. R. K. R. ENGINEERING COLLEGE**

**CHINA AMIRAM, BHIMAVARAM**

**(2017-2018)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**S. R. K. R ENGINEERING COLLEGE**

**(Affiliated to Andhra University)**

****

**CERTIFICATE**

This certify that the project titled **“CROP MANAGEMENT SYSTEM”** is the work of **Miss. P. Geeta Sri (315175710175) Miss. P. Kalyani (315175710176) Mr. P. Raj Kiran (315175710165)** who carried out the work in O**bject-Oriented Software Engineering lab** and submitted during the year 2017-2018.

**Lecturer-in-charge:**

Mr.V.Chandra Shekar Mr. Dileep Kumar

Assistant Professor, Assistant professor

Department of CSE, Department of CSE,

SRKR Engineering College, SRKR Engineering College,

Bhimavaram. Bhimavaram.

**ACKNOWLEDGEMENT**

Initially I must be thankful to persons who supported us in all aspects of my project. I convey my sincere thanks to **Mr.V.Chandra Shekar**, Assistant professor, and Mr.Dileep Kumar Assistant professor, Computer Science and Engineering, S.R.K.R Engineering College, Bhimavaram for their kind co-operation in the completion of work.

I express my heartfelt thanks to **Prof. G.V PADMA RAJU,** Head of the Department of computer science and engineering. S.R.K.R Engineering College, Bhimavaram for providing an initiative to this project and giving valuable timely suggestions for the work.

I express my sincere thanks to **Dr. G.P.S VARMA**, principal, S.R.K.R Engineering College, Bhimavaram for giving me the opportunity for the successful completion of my degree.

I also other teaching and non-teaching staff for their assistance and help extended. I thank one and all that have contributed directly or indirectly to my project.

Project Associates

315175710175

315175710176

315175710165

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No** | **CONTENTS** | **Page No** |
| **1.** | **ABSTRACT** | **7** |
| **2.** | **SOFTWARE REQUIREMENT SPECIFICATION** | **8** |
| 2.1. Introduction | **8** |
| 2.2. Existing System | **9** |
| 2.3. Proposed System | **9** |
| **3.** | **UML DESIGN** | **12** |
|  | 3.1. Use Case Diagram | **14** |
|  | 3.2. Class Diagram | **15** |
|  | 3.3. Sequence Diagram | **16** |
|  | 3.4. Collaboration Diagram | **17** |
|  | 3.5. State Chart Diagram | **18-20** |
|  | 3.6. Activity Diagram | **21-24** |
| **4.** | **SYSTEM DESIGN DOCUMENT** | **25** |
|  | 4.1. Database Design | **26** |
| **5.** | **IMPLEMENTATION SCREENS** | **27-33** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Name of the figure** | **Figure no** | **Page no** |
| **1** | Use Case Diagram | 3.1 | 14 |
| **2** | Class Diagram | 3.2 | 15 |
| **3** | Sequence Diagram | 3.3 | 16 |
| **4** | Collaboration Diagram | 3.4 | 17 |
| **5** | State chart Diagram | 3.5.1,3.5.2,3.5.3 | 18-20 |
| **6** | Activity Diagram | 3.6.1,3.6.2,3.6.3 | 22-24 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Name of the table** | **Table no** | **Page no** |
| **1** | Advisor | 4.1.1 | 25 |
| **2** | Problem | 4.1.2 | 25 |
| **3** | Solution | 4.1.3 | 26 |

**LIST OF TABLES**

1. **ABSTRACT**

Now-a-days farmers are facing many problems in agriculture due to lack of proper guidance. Some of the problems are,

* Which crop is suitable for certain season.
* The amount the fertilizers should be used.
* Suitable crop for specific soil.
* Water quantity.
* What crop will give high yield in the specific region.

In order to overcome such problem we introduce an application named **CROP** **MANAGEMENT** **SYSTEM** which is system is a suitable application for personal/individual farmers. This application is used to store the information about crops and their details.

* The advantage of this application is used to provide good suggestions over certain problems that are mentioned above.
* In this application farmers can post their problems and get suggestions based on weather conditions and type of crops grown etc

**2.SOFTWARE REQUIREMENT SPECIFICATION**

**2.1.INTRODUCTION:**

Crop Management system is the application software in which farmer can get suggestions for the good production of their crops. Farmers can post their problems like details about the crop, type of crop to be grown etc and get best solution over it. This application is used to store information about crops and their details.

**2.1.1Purpose of the system:**

The purpose of the system is mainly to get good suggestions and get a fruitful result to the farmers. In this the solutions are given according to the region and its climatic conditions. All varieties of crops and their sustainable conditions are available.

**2.1.2 Scope of the system:**

The scope of the system can be as follows

* Maintain details about the crops
* Maintain records about problems and solutions
* Provide solutions to the problems posted by the farmers.

**2.1.3 Objective:**

The objective of the application is to develop a system using which farmers can save their time, reduce investments on crops and to provide solutions faced by farmer in agriculture.

**2.2. EXISTING SYSTEM:**

In the existing system like already existing website provide information about problems and solutions faced by farmers. But they didn’t have direct communication with the farmer and advisors without authentication.

**2.3. PROPOSED SYSTEM:**

The proposed system i.e., our website have well established communication between farmer and experts without any authentication of the farmer. It provides solutions to the farmer at any time. It also provides solution based on the type of crops, wheather condtion.

**2.3.1 Overview:**

In the proposed system generally farmer interacts with the advisor through posting of problems and advisor views the problem and suggests the solution based on problem.

**2.3.2 Functional Requirements:**

The functional requirements describe the interaction between the system and its environment independent of its implementation. The environment includes the student, admin and advisor with which the system interacts.

The system consists of three modules:

* Farmer

Post the problems

View the solution for the respective problem

* Advisor

Login the system

View the problem posted by the farmer

Suggest the solution for the problem

* Admin

Login the system

Place the solution on the website

Update and maintain the data

**2.3.3 Non-Functional Requirements:**

**Safety Requirements:**

It is required to take the database backup within a particular time.

**Response time:**

The system gives feedback to the user in a certain minimum time.

**Quality Issues:**

* In the event of failure, the system cannot loss any data.
* System can handle exception such as invalid contents, incomplete field.
* Secure access of confidential data.

**System Modifications:**

The system is flexible, such that any further modifications can be done.

**Reliability:**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by java is more reliable.

**Cost:**

The cost of development and maintenance is low. As this project uses all open source tools to develop, the cost of construction is low.

**Availability:**

Amount of time that a server is running and available to respond users.

**Security:**

This avoids the non-authorised users’ access to the website.

**2.3.4 Hardware Requirements:**

Processor : Intel Pentium III or equivalent

RAM : 512MB

Cache Memory : 512KB

Hard disk : 20GB

**2.3.5 Software Requirements:**

UML : Rational Rose

Web technology : HTML, Java Script

Database : MYSQL

Operating system : Windows XP, Windows 7

**3. UML DESIGN**

The Unified Modeling Language (UML) is a general purpose visual modeling language that is used to specify, visualize, construct, and document the artefacts of a software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. The Unified Modeling Language is very important parts of developing object- oriented software and the software development process. The Unified Modeling Language uses mostly graphical notations to express the design of software projects. Using the Unified Modeling Language helps project teams communicate, explore potential designs, and validate the architectural design of the software. The primary goals in the design of the Unified Modeling Language are:

* Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models. Provide extensibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development processes.
* Provide a formal basis for understanding the modeling language. Encourage the growth of the object oriented tools market.
* Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
* Each Unified Modeling Language diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction.

UML diagrams commonly created in visual modeling tools include.

* Use Case Diagram
* Class Diagram
* Sequence Diagram
* Collaboration Diagram
* State Chart Diagram
* Activity Diagram

**3.1 USE CASE DIAGRAM:**

A use case diagram in the UML is a type of behavioural diagram defined by and created from a use-case analysis. The purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those two use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

**3.1.1 Identification of Actors:**

Actors represent system users. They help delimit the system and give a clearer picture of what the system should do. It is important to note that an actor interacts with, but has no control over the use cases.

An actor is someone or something that:

* Interacts with or uses the system.
* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

An actor can be represented as shown below:

1. A class rectangle with the label

|  |
| --- |
| <<actor>>  Administrator |

1. The stick figure with the name of the actor below.

User

Administrator

Actors identified are:

1. Administrator.

2. User.

**3.1.2 Identification of Use cases:**

In its simplest form, a use case can be described as a specific way of using the system from a user’s (actors) perspective.

A more detailed description might characterize a use case as:

* A pattern of behaviour the system exhibits.
* A sequence of related transactions performed by an actor in the system.
* Delivering something of value to the actor.

Use cases provide a means to:

* Capture system requirements.
* Communicate with the end users and domain experts.
* Test the system.

**3.1.3 Constructions of Use case diagrams:**

Use-case diagrams graphically depict the system behaviour (use cases).These diagram’s present a high level view of how the system is used as viewed from an outsider’s(actor’s) perspective.

A use-case diagram can contain:

* Actors (“things” outside the system)
* Use cases (system boundaries identifying what the system should do).
* Interactions or relationships between actors and use cases in the System include the associations, dependencies, and generalizations.

**3.1.4 Uses Association:**

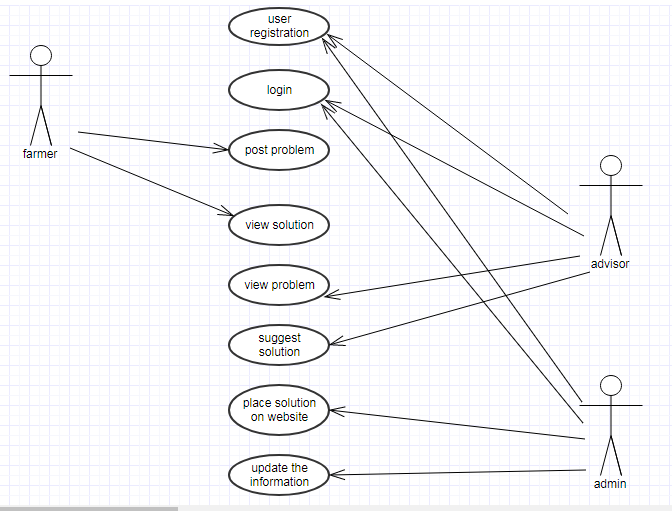
The uses association occurs when we are describing our use-cases and notice that some of them have sub flows in common. To avoid describing a sub flow more than once in several use-cases, you can extract the common sub flow and make it a use-case of its own.

**3.1.5 Extends Association:**

An Extends association is a stereotyped association that specifies how the functionality of one of use case can be inserted into the functionality of another use-case.

**3.1.6 Includes Association:**

An Includes association is a stereotyped association that connects a base use case to an inclusion use-case.

****

3.1.Use Case Diagram

**3.2 CLASS DIAGRAM:**

A class diagram shows a set of classes, interfaces, collaborations and their relationships. The diagrams are most common diagram founding modelling object-oriented systems. Class diagram is a collaboration of attributes, operations, objects, classes and relationships among the classes. It represents overall structure of project. It is the main diagram for overall project.

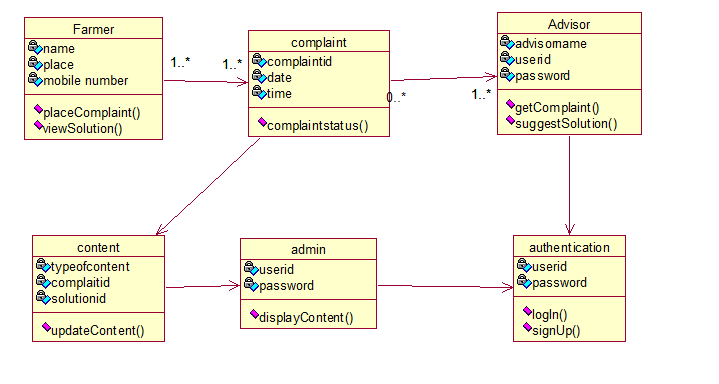
**3.2.1 Identification of Attributes of each class:**

Guide lines for identifying attributes of classes are as follows:

* Attributes usually correspond to nouns followed by prepositional phrases. Attributes also may correspond to adjectives or adverbs.
* Keep the class simple State only enough attributes to define the object state.
* Attributes are less likely to be fully described in the problem statement. Omit derived attributes.
* Do not carry discovery attributes to excess.

**Identification of Relationships of each class:**

**Association:** This relationship represents a physical or conceptual connection between two or more objects.



3.2 Class Diagram

**3.3 SEQUENCE DIAGRAM:**

The dynamic model represented in UML with interaction diagrams, state chart diagrams, and activity diagrams, describes the internal behaviour of the system.

**Construction of Sequence Diagrams:**

* A Sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next.
* Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

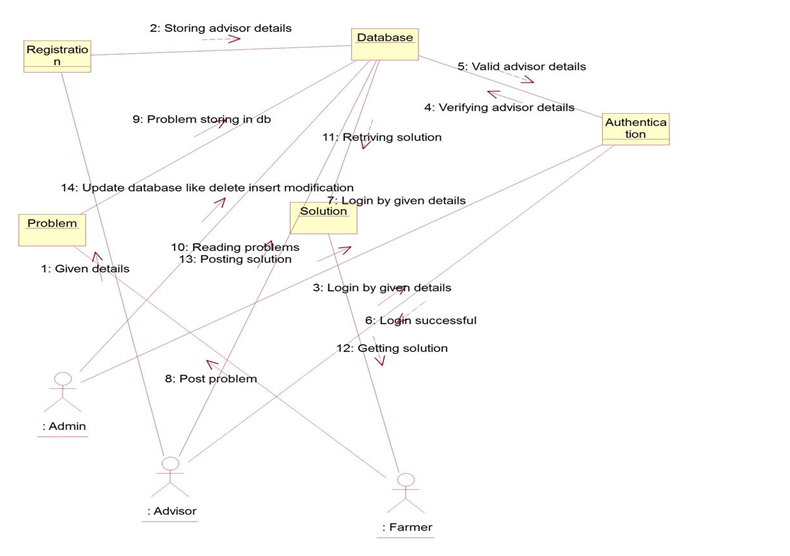
****

3.3 Sequence Diagram

**3.4COLLABORATION DIAGRAM:**

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time based sequence.

Collaboration diagrams show objects, their links and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.



3.4 Collaboration Diagram

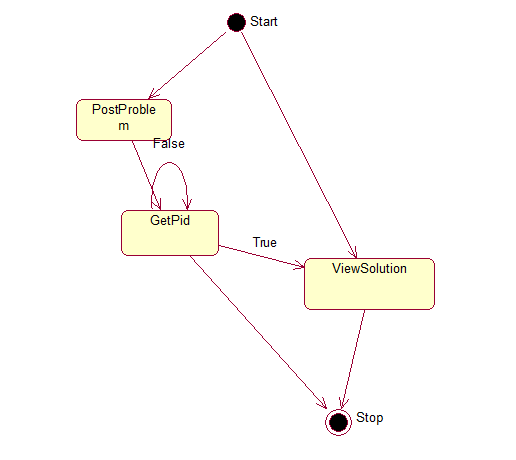
**3.5 STATE CHART DIAGRAM:**

State chart diagrams model the dynamic behaviour of individual classes or any other kind of object. They show the sequences of states that an object goes through the events that cause a transition from one state to another and the actions that result from a state change. State chart diagrams are closely related to activitydiagrams. The main difference between the two diagrams is state chart diagrams are state centric, while activity diagrams are activity centric. A state chart diagrams are activity centric. A state chart diagram is typically used to model the discrete stages of an object’s lifetime, where as an activity diagram is better suited to model the sequence of activities in a process. Each state represents a named condition during the life of an object during which it satisfies some conditions or waits for an event.

The following tools are used on the state chart diagram toolbox to model state chart diagrams:

* **Decisions**: A decision represents a specific location on state chart diagram where the work flow may branch based upon guard conditions.
* **Synchronization**: Synchronizations visually define forks and joins representing parallel work flow.
* **Forks and Joins**: A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows. A join consists of two of more flows of control that unite into single flows of control that unite into single flow of control.
* **States:** A state represents a condition or situation during the life of an object during which it satisfies some conditions or waits for some event.
* **Transitions**: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.
* **Start states**: A start state (also called an “initial state”) explicitly shows the beginning of a work flow.

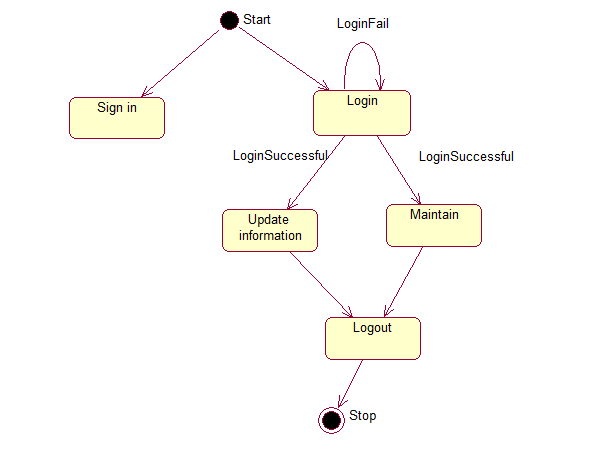
**End States**: An end state represents a final state or terminal state.

****

3.5.1 State Chart Diagram for farmer



3.5.2 State Chart Diagram for Advisor



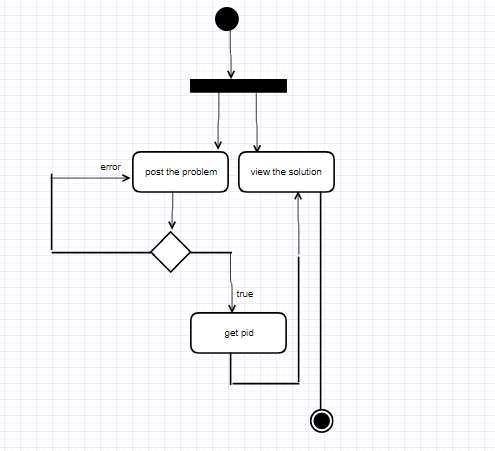
3.5.3State Chart Diagram for Advisor

**3.6 ACTIVITY DIAGRAM:**

Activity diagrams provide a way to model the workflow of a business process. An activity diagram is typically used for modelling the sequence of workflows. A software company could use activity diagrams to model a software development process.

The following tools are used on the activity diagram toolbox to model activity diagrams:

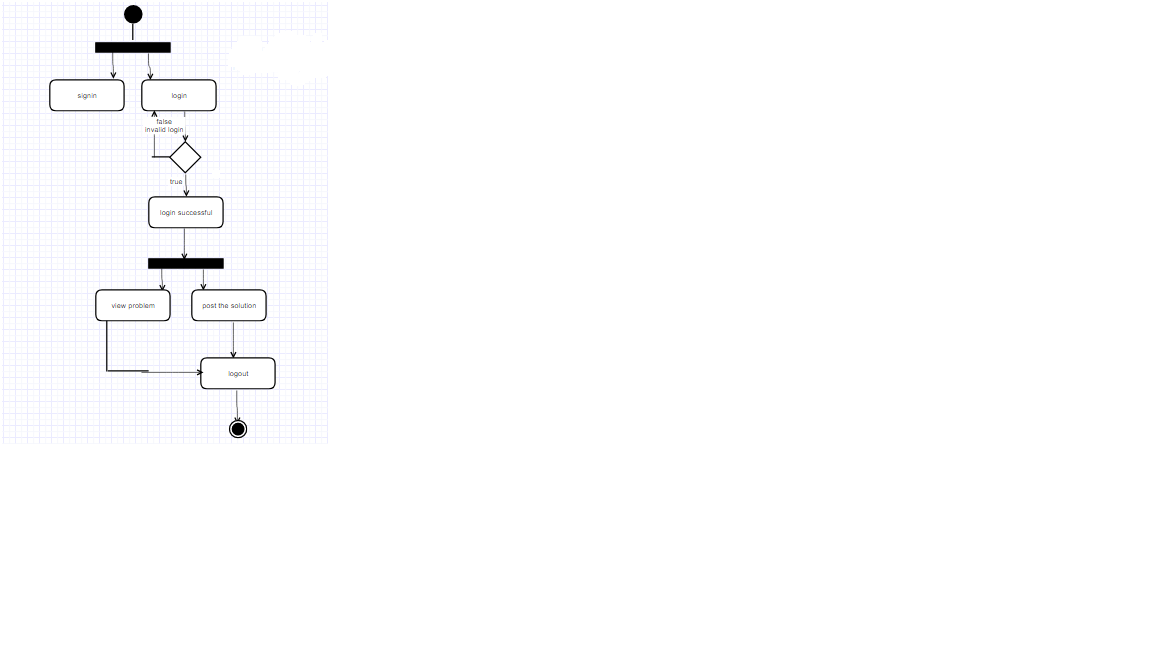
* **Decisions**: A decision represents a specific location on activity diagram where the workflow may branch based upon guard conditions.
* **Synchronizations**: Synchronizations visually define forks and joins representing parallel workflow.
* **Forks and Joins**: A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows. A join consists of two or more flows of control that unite into a single flow of control.
* **States**: A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event.
* **Transitions**: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.
* **Start states**: A start state (also called “initial state”) explicitly shows the beginning of a workflow.
* **End States**: An end state represents a final state or terminal state.
* **Swim lane:** A unique activity diagram feature that defines who or what is responsible for carrying out the activity or state.



3.6.1 Activity diagram for Farmer

 3.6.2 Activity Diagram for admin

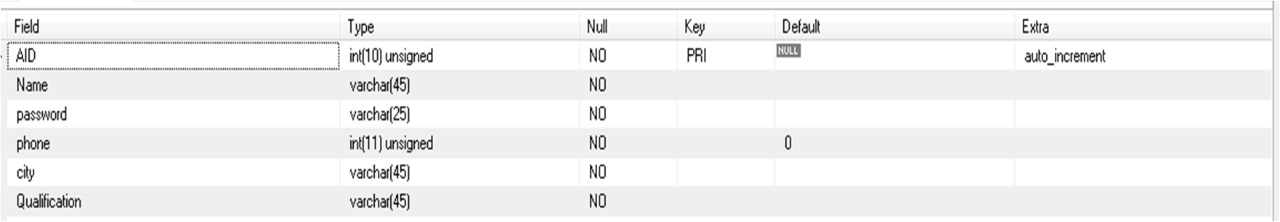
3.6.3Activity diagram for Advisor

**4. SYSTEM DESIGN DOCUMENT**

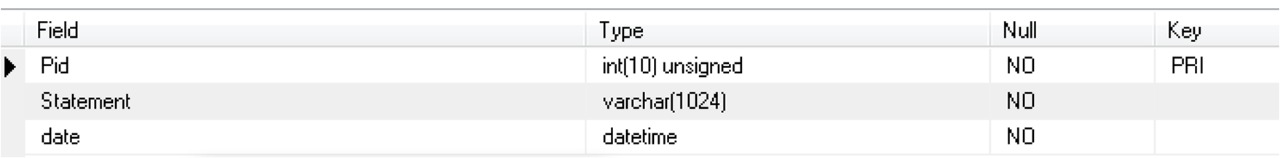
**4.1. DATABASE DESIGN:**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. It can be thought of as the logical design of the base data structures used to store the data.

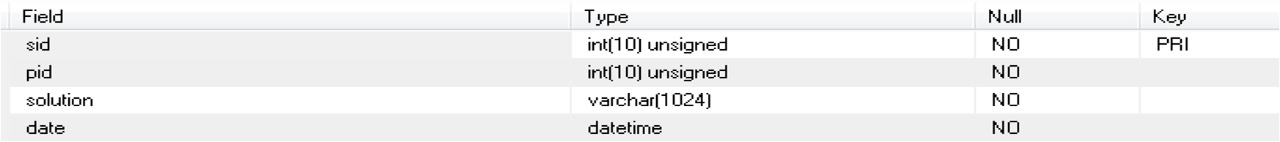
**Table 4.1.1** Advisor



**Table 4.1.2 Problem**

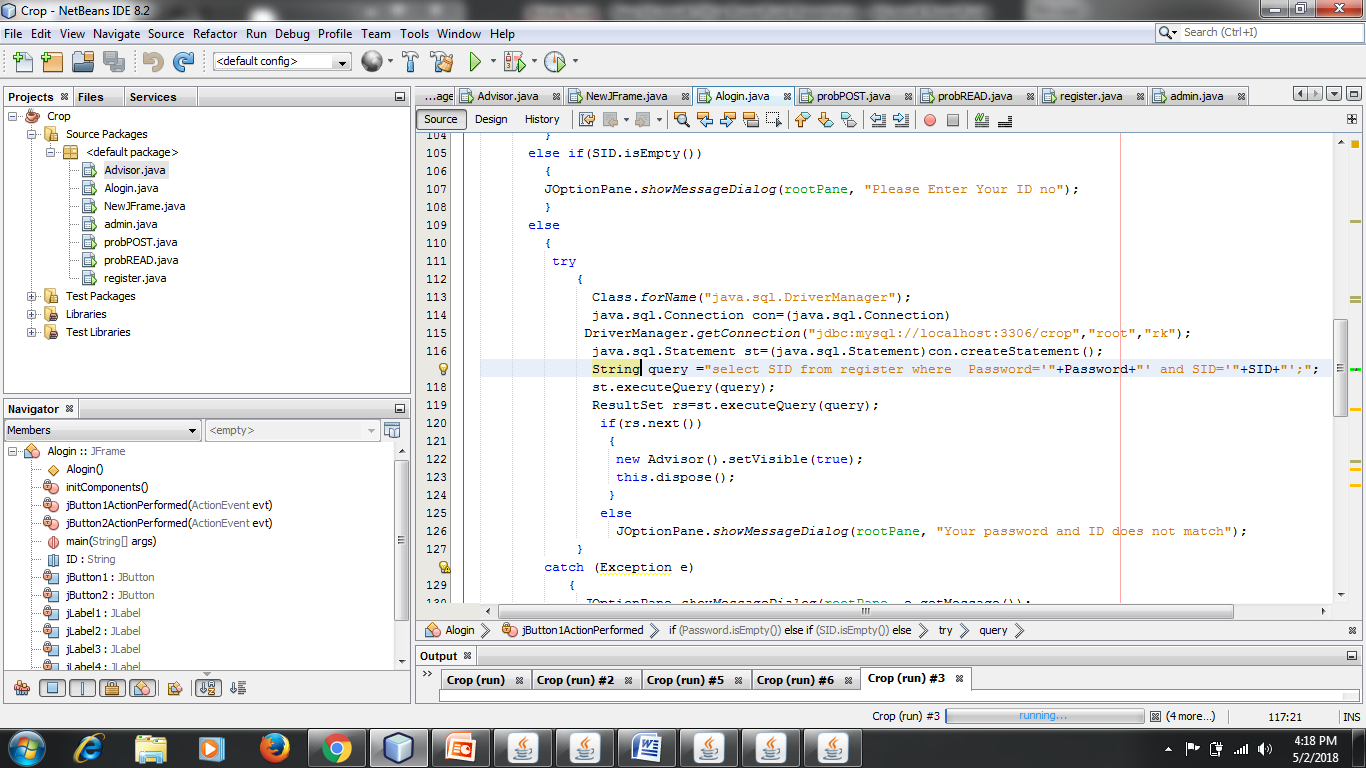
****

**Table 4.1.3 Solution**

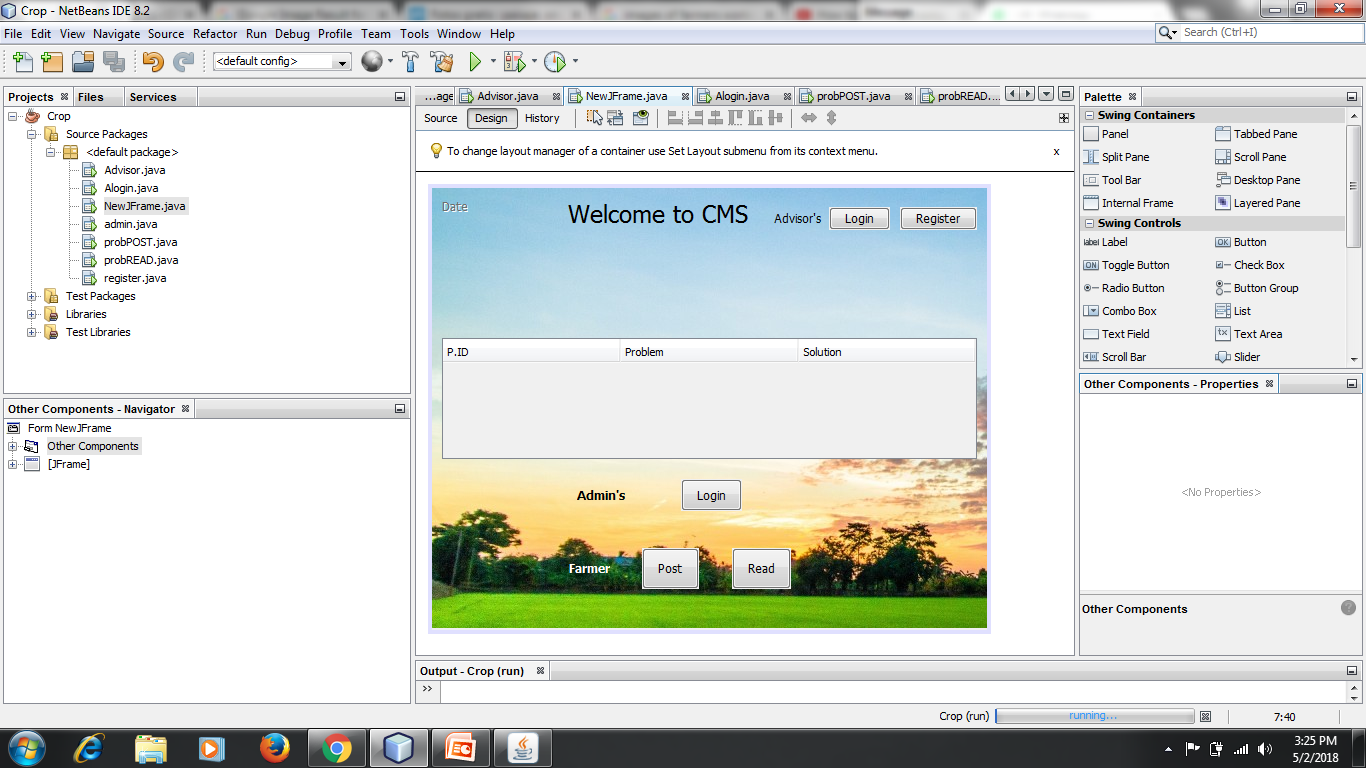
****

**5. IMPLEMENTATION SCREENS**

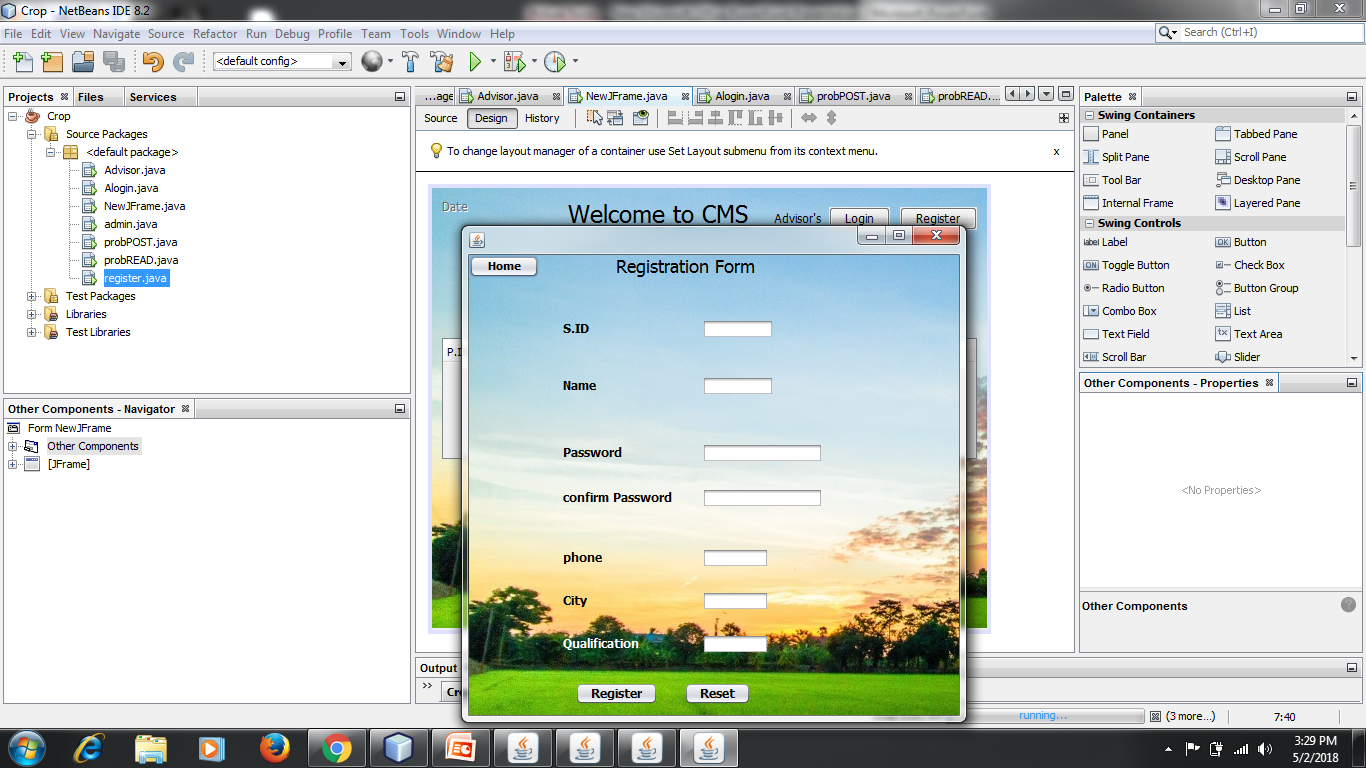
**The following image tells us that the required frames for this project**

****

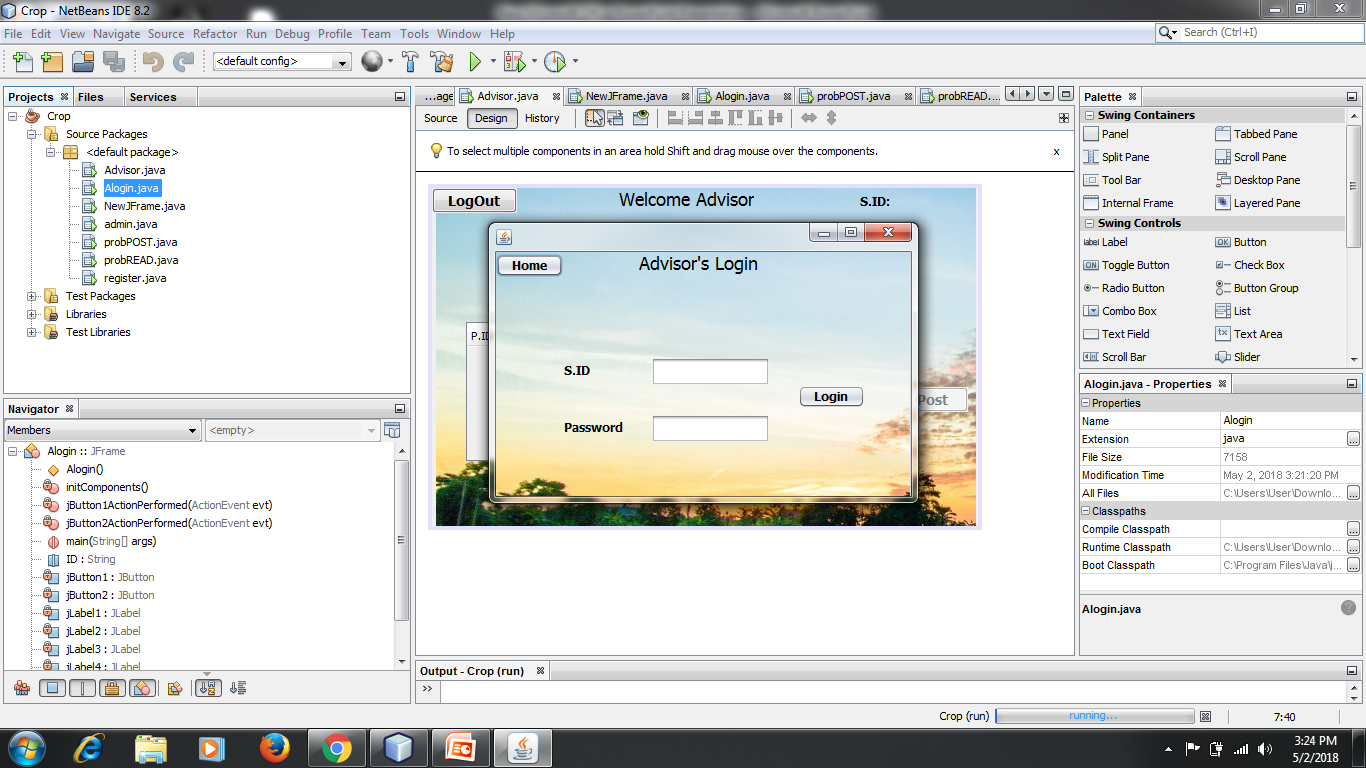
**Home page**

****

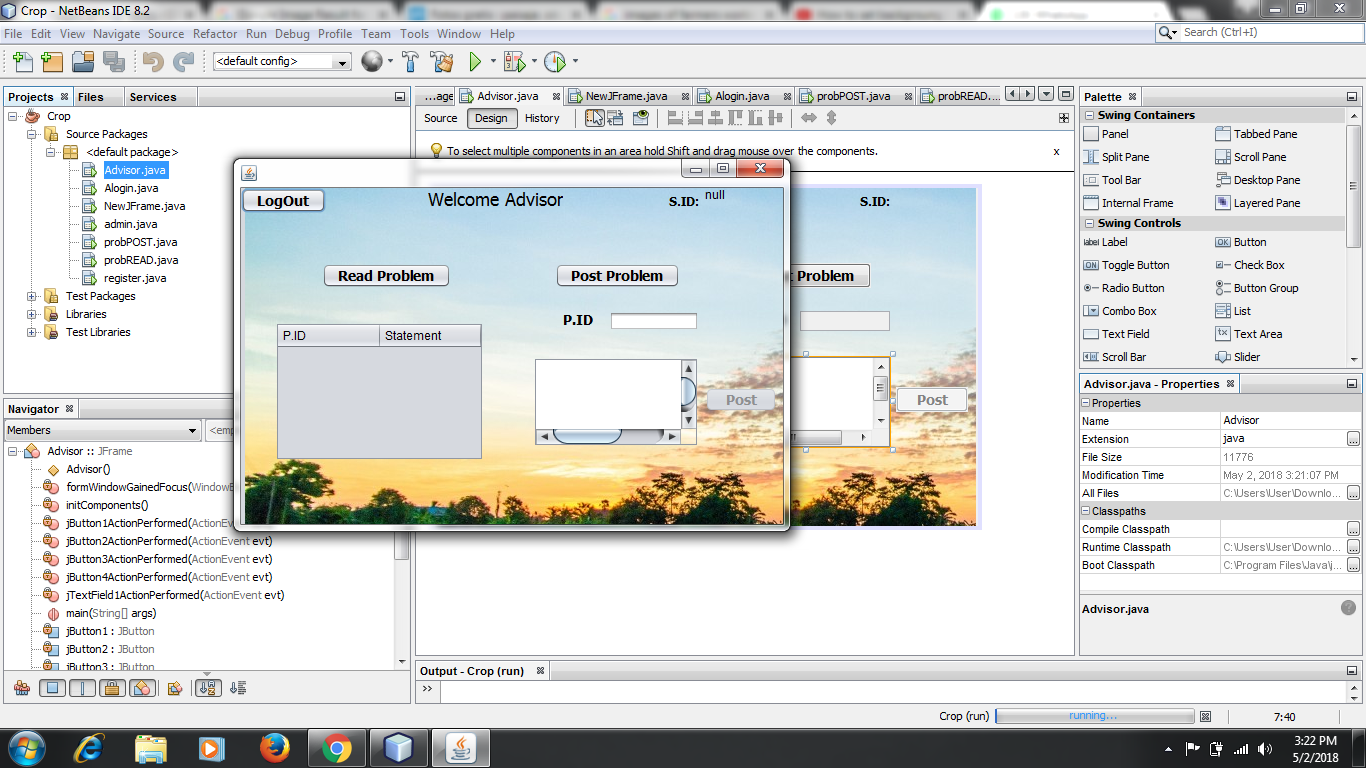
**Advisor’s registration page**

****

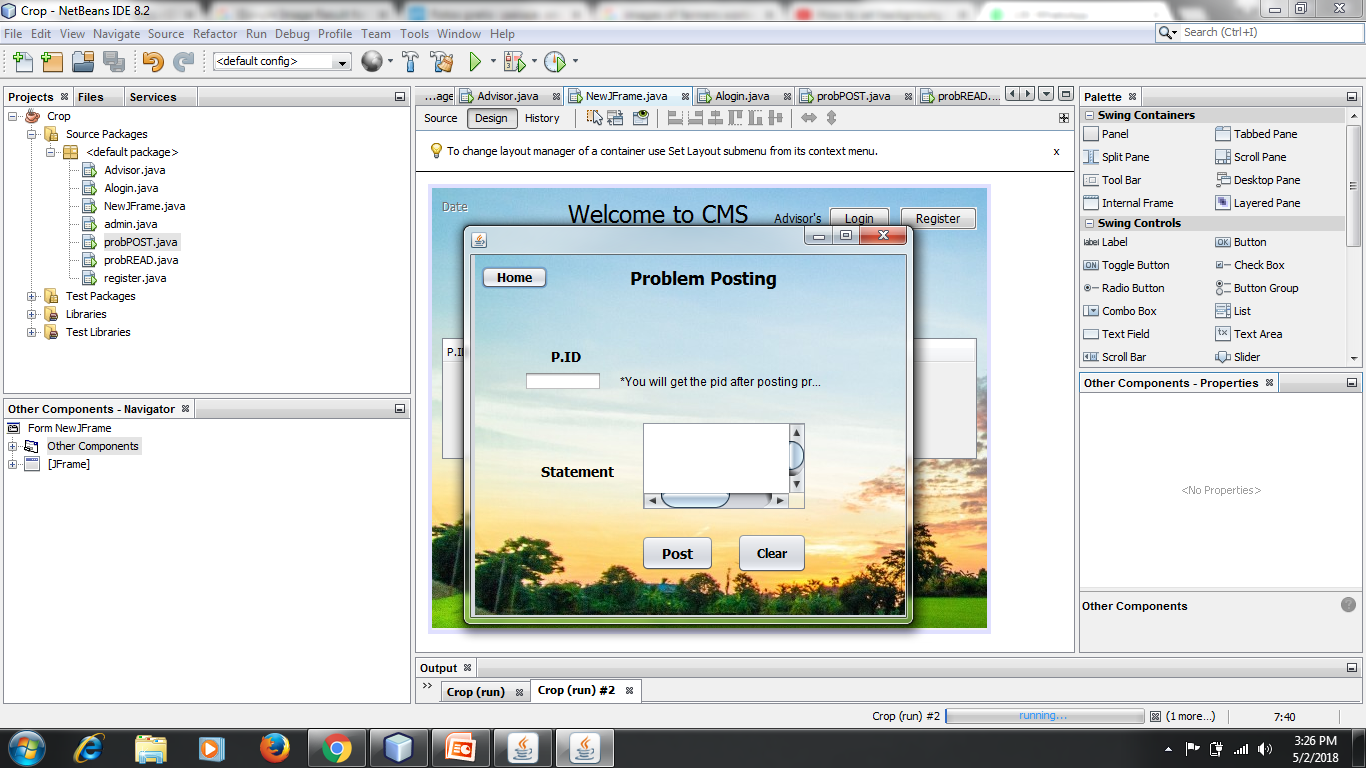
**Advisor’s login page**

****

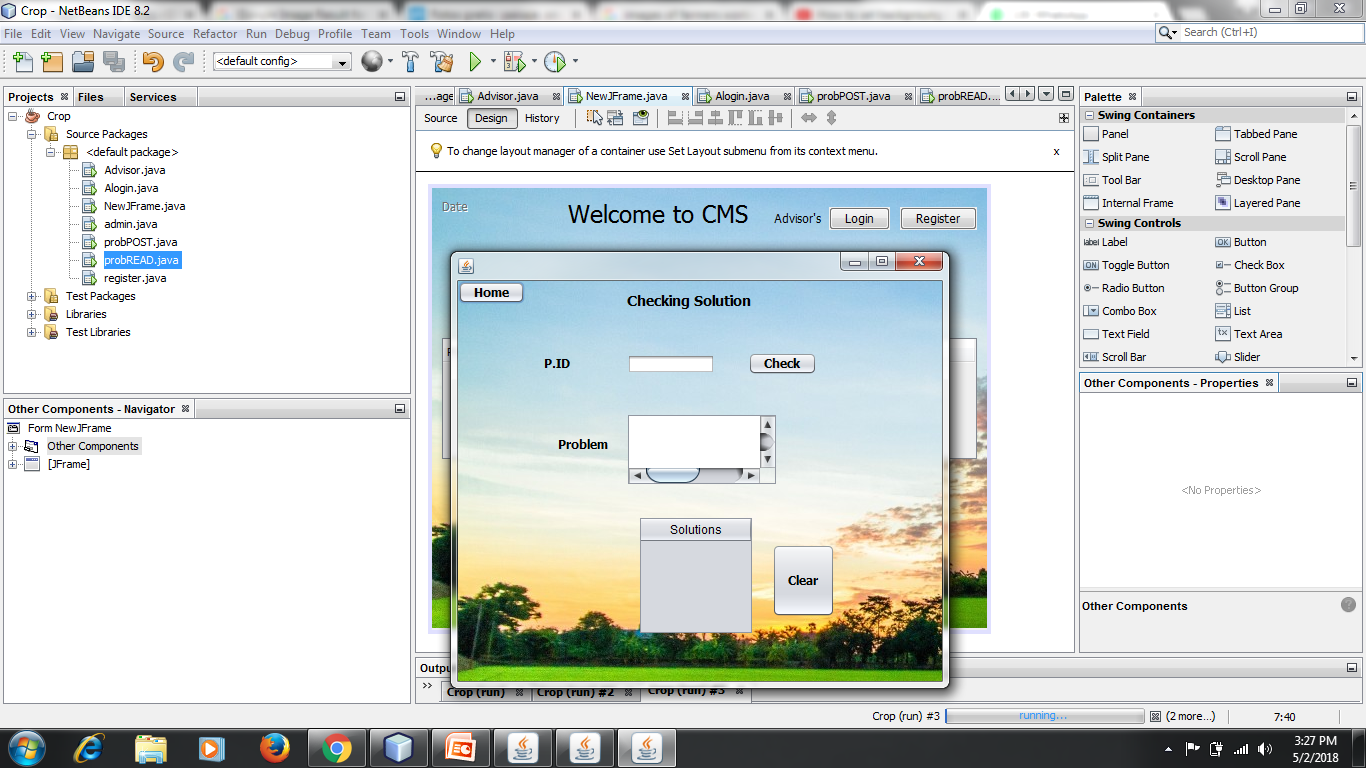
**Advisor’s page**

****

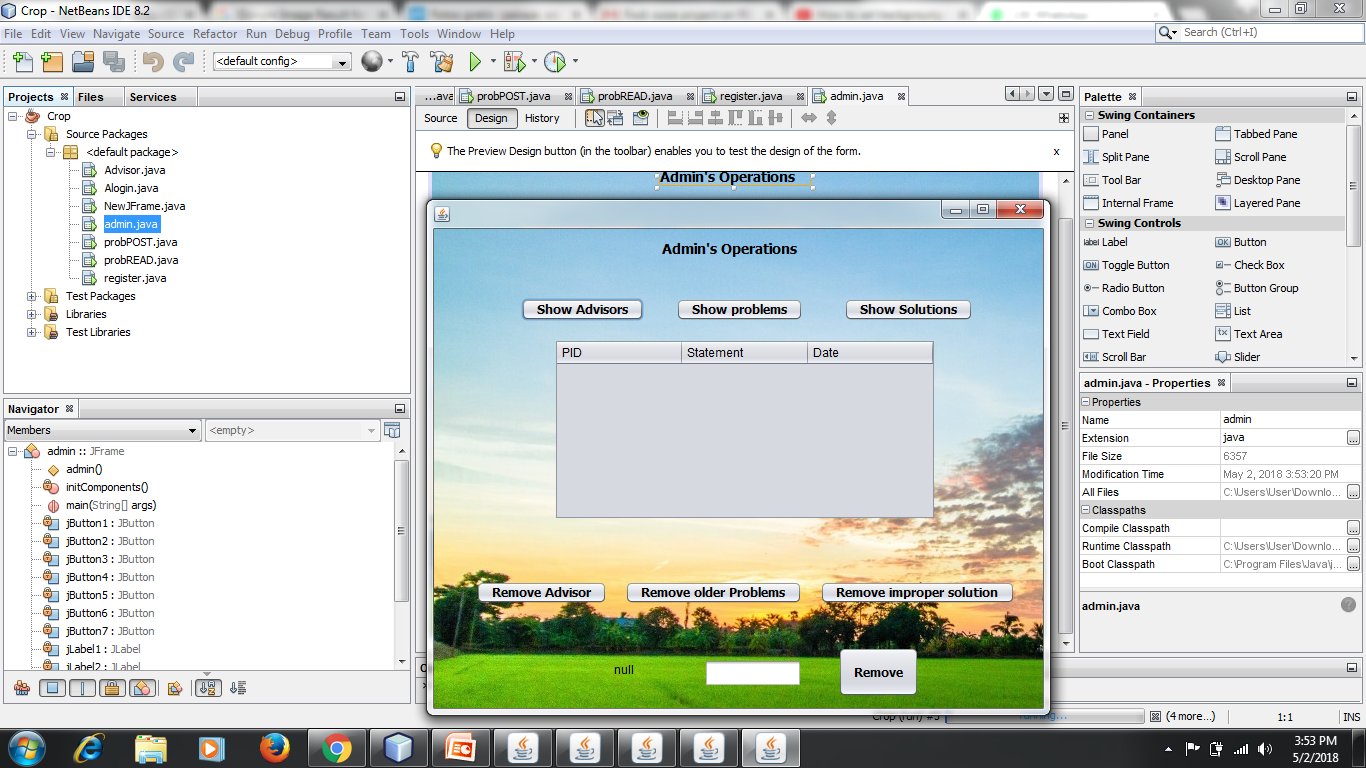
Problem posting page

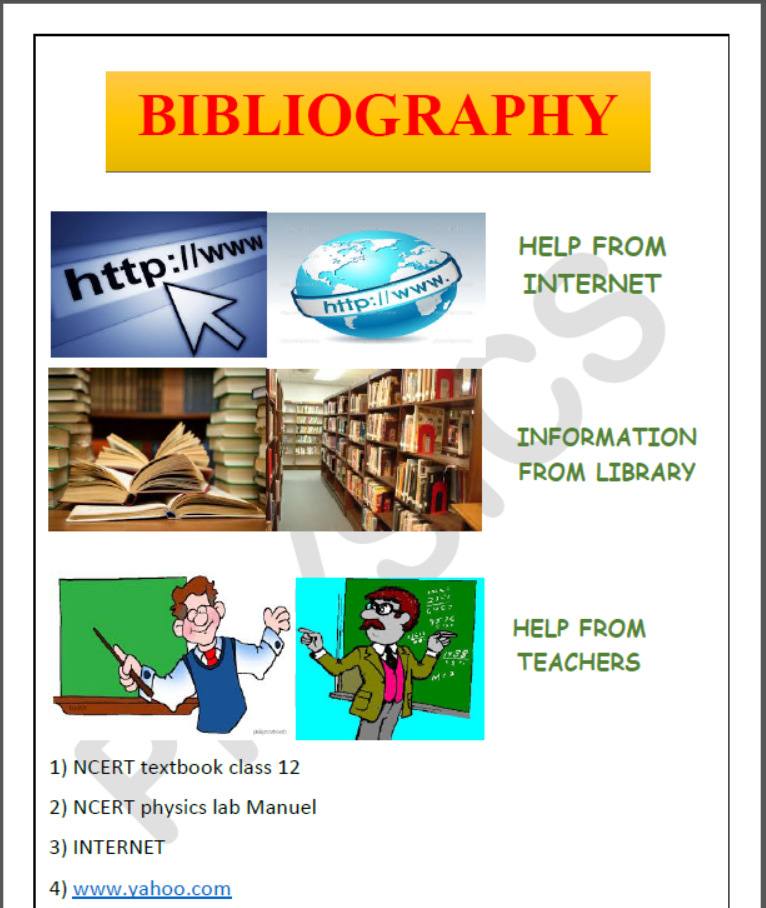


Solutions page

****

Admin’s page





1. Object oriented software engineering by Allen H. Dutoit & Bernd Bruegge.

2. Web Technology and Design by C. Xavier.

3. [www.google.com](http://www.google.com)

4. [www.w3schools.com](http://www.w3schools.com)

**THANK YOU**