PROJECT REPORT ON

VEHICLE DETECTION AND SPEED TRACKING USING MACHINE LEARNING

Submitted to Kakatiya University in Partial fulfillment for the Award of Degree

MASTER OF COMPUTER APPLICATIONS

By

GADDELAPALLI MANOJ CHARY (22162I0010)

Under the Guidance of

SHEEMA SABAHATH

Assistant Professor

VAAGDEVI COLLEGE OF ENGINEERING(MCA)

(Affiliated to Kakatiya University, Approved by AICTE)

P.O.BOLLIKUNTA, WARANGAL URBAN-506 005(T.S)

2022-23

VAAGDEVI COLLEGE OF ENGINEERING(MCA)

(Affiliated to Kakatiya University, Approved by AICTE)

P.O.BOLLIKUNTA, WARANGAL URBAN-506 005(T.S)

MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the major project entitled **Vehicle Detection And Speed Tracking** is submitted by **Gaddelapalli Manoj Chary** bearing ***22162I0010*** in partial fulfillment of the requirements for the award of the Degree in Master Of Computer Applications during the academic year 2022-2023.

Guide **Head of the Department** **External Examiner**

ACKNOWLEDGEMENT

We would like to express our gratitude to all the people behind the screen who helped us to transform idea into a real application.

We would like to thank our Director **Dr.Ch.Vani,** for her encouragement and valuable guidance in bringing shape to this discussion

We Profoundly thank **Dr K.Rajesh Khanna**, Head of the Department of Master of Computer Applications who has been excellent guide and also a great source of inspiration to our work.

We would like to thank our guide **Sheema Sabahath**, Assistant Professor for providing the excellent facilities, motivation and valuable guidance throughout the project work. With her co-operation and encouragement, we completed the project work in time.

We also extend my thanks to my faculty of **VAAGDEVI COLLEGE OF ENGINEERING (MCA)** for their support during my course.

With Sincere Regards

GADDELAPALLI MANOJ CHARY (22162I0010)

ABSTRACT

In recent times, there has been a drastic change in people's lifestyles and with an increase in incomes and lower cost of automobiles there is a huge increment in the number of cars on the roads which has led to traffic and commotion. The manual efforts to keep people from breaking traffic rules such as the speed limit are not enough. There is not enough police and man force available to track the traffic and vehicles on roads and check them for speed control. Hence, we require technologically advanced speed calculators installed that effectively detect cars on the road and calculate their speeds.

To implement the above idea two basic requirements, need to be met which are the effective detection of the cars on roads and their velocity measurement. For this purpose, we can use OpenCV software which uses the Haar cascade to train our machine to detect the object, in this case the car.

we have developed a Haar cascade to detect cars on the roads, whose velocities are then measured using a python script. The real-time application of this project proves to be much useful as it is easy to implement, fast to process and efficient with low-cost development. Also, the tool might be useful to apply in simulation tools to measure velocities of cars. This can be further developed to identify all kinds of vehicles as well as to check anyone who breaks a traffic light.

The improvements in the project can be done by creating a bigger haar cascade since bigger the haar cascade developed, more the number of vehicles that can be detected on the roads. Better search algorithms can allow a faster search and better detection of these vehicles for better efficiency.

INDEX

1. INTRODUCTION

The continuously increasing number of on-road vehicles has put a lot of pressure on road capacity and infrastructure, making traffic management difficult and giving way to problems like congestion, collisions, and air pollution, among others. These problems have significant impact on our daily lives. A robust and efficient traffic management system is required to reduce their effect. A large amount of traffic data is generated daily. Traffic data contains information related to traffic flow, distribution, pattern, and collisions, which can be used to solve various traffic related issues. The volume and distribution of traffic can be used to build appropriate structural and geometric designs for road segments. Traffic collisions can be analyzed to see the correlation of traffic volume and number and severity of collisions, which in turn can help to investigate and assess collision risks. Apart from these problems related to vehicle traffic, the data can also help in studies related to the reduction of environment pollution and fuel consumption. Also, various statistical parameters, such as the average number of vehicles on the road at a certain time, and the state of congestion, can be studied, which can provide some information for managing the highway [17]. To address these pressing issues, NVIDIA initiated a series of challenges aimed at the intersection of Artificial Intelligence and Smart City. The first AI City Challenge, organized in 2017, focused on object detection, localization, and classifications. The 2018 AI City Challenge continues to promote deep learning and computer vision approaches to help analyze urban traffic videos, and finally improve traffic conditions and prevent traffic collisions. This workshop especially focuses on solving serious problems related to urban traffic. Specifically, the challenge is comprised of three tracks: Track1 (Traffic Flow Analysis) aims at developing models that can predict the driving speed of vehicles in videos recorded by stationary cameras on highways or at intersections. Track2 (Anomaly Detection) focuses on detecting driving anomalies, such as stalled vehicles or car accidents, in the videos. Track3 (Multi-camera Vehicle Detection and Re-identification) aims to detect and track all vehicles in a set of videos which appear in each

of a subset of the videos recorded at different locations within the city. In this paper, we propose a formulation to solve Track1. Our model relies heavily on vehicle detection and tracking. In this Challenge, however, it is hard to train a vehicle detection model from scratch since no labeled data is provided. Instead, we leverage transfer learning and perform inference on our dataset using the 3D Deformable model for vehicle detection. As an alternative to the 3D Deformable model, we considered the model by Bhandary et al, which obtained similar performance in the 2017 challenge. To be able to compare these two models on the target reference data, we extract all frames from one of the Track 1 videos and measure the models' performance on the frames, comparing the vehicle detection performance as measured by mean Average Precision (mAP). The experiment shows that the 3D Deformable model achieves 74% mAP, which is higher than the model of Bhandary et al. The methodology of our tracker is detect-then-track. The performance of the tracker is thus highly dependent on the accuracy of the detection. For each frame, we extract salient features from the detected vehicles and then identify them in the next frame. The change of in-frame location of these features contributes the necessary information for estimating the vehicle's speed.

1. LITERATURE SURVEY

Vehicle tracking is required in order to build a robust vehicle speed estimation model. Many methods have been developed that use classic computer vision and machine learning approaches for object tracking. Kale et al. (2015) utilized a classic optical-flow algorithm as well as motion vector estimation to solve the object tracking problems. They proposed a track-by-detect approach, where detection was done by using an optical-flow algorithm and speed estimation was handled by motion vector estimation. Geist et al. (2009) contributed a reinforcement learning-based framework, combined with a Kalman Filter to address the non-stationary environment. The paper proposes that tracking objects in the video can be viewed as the problem of predicting the location of the bounding box of that targeted object at each frame. Zhang et al. (2017) developed a recurrent convolutional neural network model trained with a Reinforcement Learning algorithm. Faragher (2012) presented a simple and detailed explanation of Kalman Filters. The paper explains the assumption behind Kalman Filters and derives the process of modeling a tracking problem mathematically, step by step. Brasnett et al. (2005) proposed an approach to track objects by combining features with a particle filtering algorithm, solving nonlinear, non-Gaussian tracking problems. Many research studies have also been conducted in the field of vehicle speed detection with various approaches. Rad et al. (2010) has proposed an approach involving the comparison of the vehicle position between the current frame and the previous frame to predict traffic speed from digital video captured with a stationary camera. The camera calibration was done by applying geometrical equations. The system designed by Rad et al. has the potential to be extended to other application domains and has an average error of ±7 km/h for the detected vehicle speed. Ferrier et al. (1994) used the motion parameters in the image, along with information on the projection between the ground plane and the image plane, to obtain various metrics, including vehicle speed, using real-time tracking techniques. They have also used scene specific tuning of the dynamics for more accurate prediction of target location by the tracker.

Yamazaki et al. (2008) have used digital aerial images to detect vehicle speed by extracting the vehicles and shadows from two consecutive images. The speed is detected by linking the corresponding vehicles from these images based on their distance, order and size and then using distance between corresponding vehicles and time lag. Wu et al. (2009) have utilized mapping of the coordinates in the image domain into the real-world domain. Liu and Yamazaki have used a pair of Quick Bird panchromatic and multi-spectral images for speed detection. Geratetal. used the combination of Kalman filters and optical flow approaches to estimate speeds. The former is helpful in avoiding the problem of temporary occlusions, while the latter provides more accurate speed delivery. Wang presented an approach based on moving target detection in a video by mapping the relation between pixel distance and actual distance. In this algorithm, three frame differencing and background differencing were used to extract features from moving vehicles. Then, tracking and positioning was done using vehicle centroid feature extraction.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In recent times, there has been a drastic change in people's lifestyles and with an increase in incomes and lower cost of automobiles there is a huge increment in the number of cars on the roads which has led to traffic and commotion. The manual efforts to keep people from breaking traffic rules such as the speed limit are not enough. There is not enough police and man force available to track the traffic and vehicles on roads and check them for speed control. Hence, we require technologically advanced speed calculators installed that effectively detect cars on the road and calculate their speeds.

3.2 PROPOSED SYSTEM

we have developed a Haar cascade to detect cars on the roads, whose velocities are then measured using a python script. The real-time application of this project proves to be much useful as it is easy to implement, fast to process and efficient with low-cost development. Also, the tool might be useful to apply in simulation tools to measure velocities of cars. This can be further developed to identify all kinds of vehicles as well as to check anyone who breaks a traffic light.

The improvements in the project can be done by creating a bigger haar cascade since bigger the haar cascade developed, more the number of vehicles that can be detected on the roads. Better search algorithms can allow a faster search and better detection of these vehicles for better efficiency.

3.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.3. ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.5 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

4. SYSTEM REQUIREMENTS SPECIFICATION

4.1 INTRODUCTION

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

4.2 PURPOSE

Typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

4.3 FUNCTIONAL REQUIREMENTS

- Graphical User interface with the User.

For developing the application, the following are the Software Requirements:

1. Python
2. anaconda

OPERATING SYSTEMS SUPPORTED

1. Windows 7
2. Windows XP
3. Windows 8

TECHNOLOGIES AND LANGUAGES USED TO DEVELOP

1. Python

DEBUGGER AND EMULATOR

- Any Browser (Particularly Chrome)

HARDWARE REQUIREMENTS

For developing the application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB

Space on Hard Disk: minimum 512MB

4.4 NON-FUNCTIONAL REQUIREMENTS

- Any Browser (Particularly Chrome)

4.5 INPUT & OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow:

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.

Confirm an action

4.6 HARDWARE REQUIREMENTS

For developing the application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher

- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

4.7 SOFTWARE REQUIREMENTS

For developing the application, the following are the Software Requirements:

3. Python
4. anaconda

OPERATING SYSTEMS SUPPORTED

4. Windows 7
5. Windows XP
6. Windows 8

TECHNOLOGIES AND LANGUAGES USED TO DEVELOP

Python

5.SYSTEM DESIGN

UML

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module are usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, where as during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

System Model

Introduction to UML

The unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct and document the artifacts of software-intensive system.

The goal of UML is to provide a standard notation that can be used by all object - oriented methods and to select and integrate the best elements .UML is itself does not prescribe or advice on how to use that notation in a software development process or as part of an object- design methodology. The UML is more than just bunch of graphical symbols. Rather, behind each symbol in the UML notation is well-defined semantics.

The system development focuses on three different models of the system.

➔ **Functional model**

➔ **Object model**

➔ **Dynamic model**

Functional model in UML is represented with use case diagrams, describing thefunctionality of the system from user point of view.

Object model in UML is represented with class diagrams, describing the structure of thesystem in terms of objects, attributes, associations and operations.

Dynamic model in UML is represented with sequence diagrams, start chart diagrams andactivity diagrams describing the internal behaviour of the system.

Scenarios

A Use Case is an abstraction that all describes all possible scenarios involving the described functionality. A scenario is an instance of a use case describing a concrete set of actions.

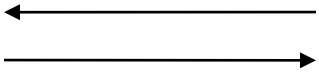
- The **Name** of the scenario enables us to refer it ambiguously. The name of scenario is underlined to indicate it is an instance.
- The **Participating actor instance** field indicates which actor instance are involved in this scenario. Actor instance also have underlined names.

➔ The **Flow of Events** of scenario describe the sequence of events step by step.

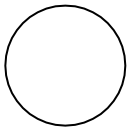
Data Flow Diagram:

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart. DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

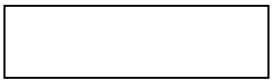
1. Dataflow: Data move in a specific direction from an origin to a destination.



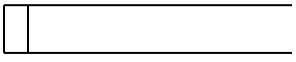
2. Process: People, procedures, or devices that use or produce (Transform) Data. Thephysical component is not identified.



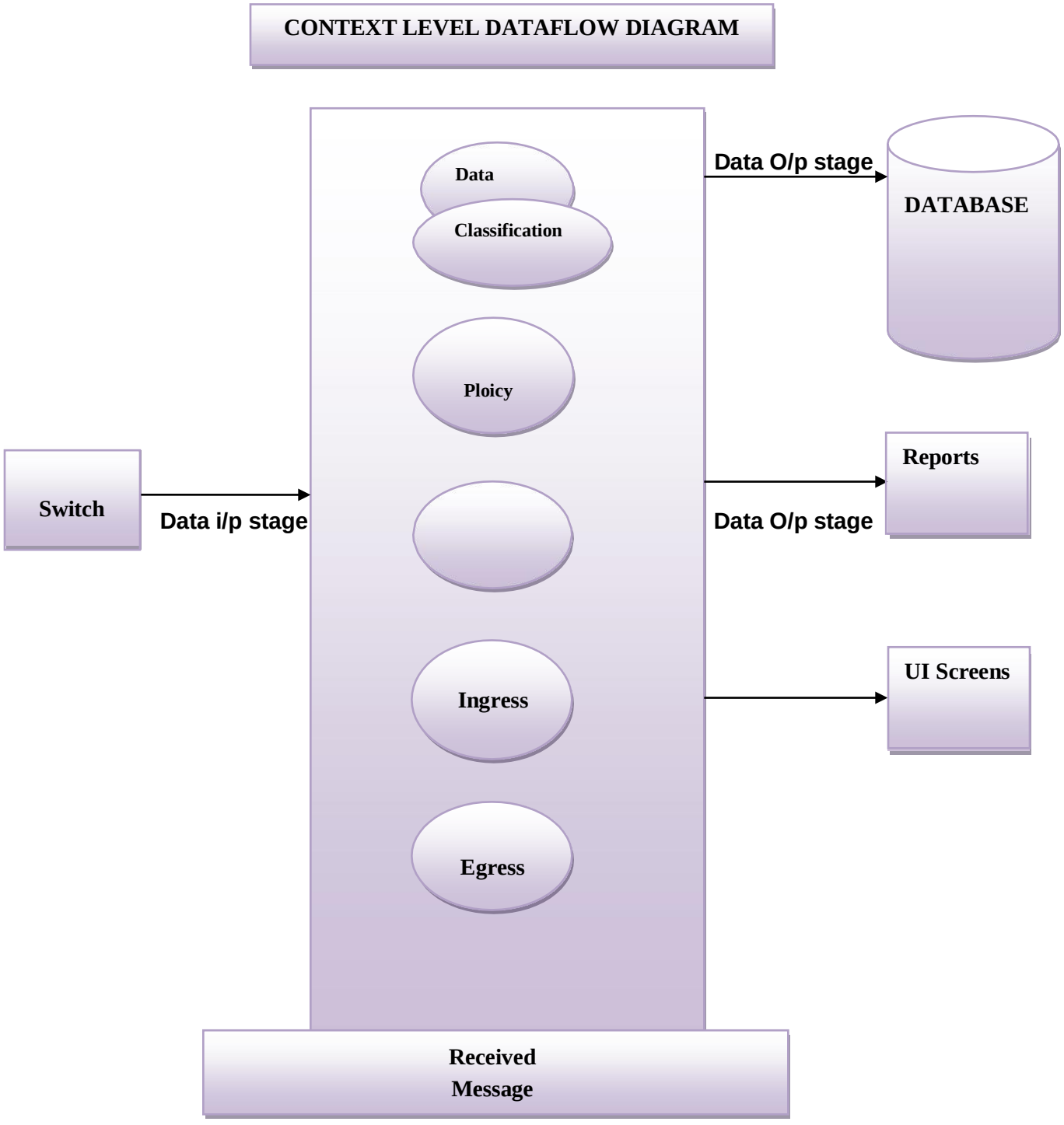
3. Source: External sources or destination of data, which may be People, programs,organizations or other entities.



4. Data Store: Here data are stored or referenced by a process in the System.



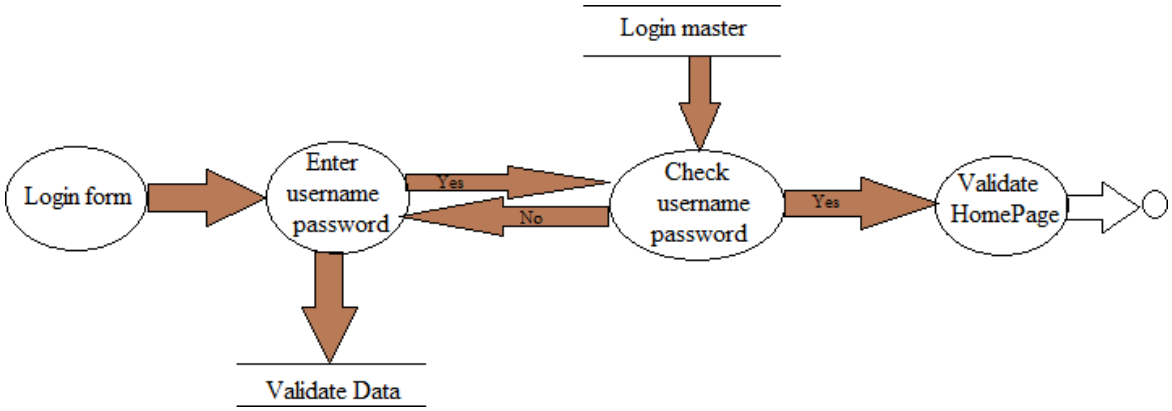
CONTEXT LEVEL 0 DIAGRAM:



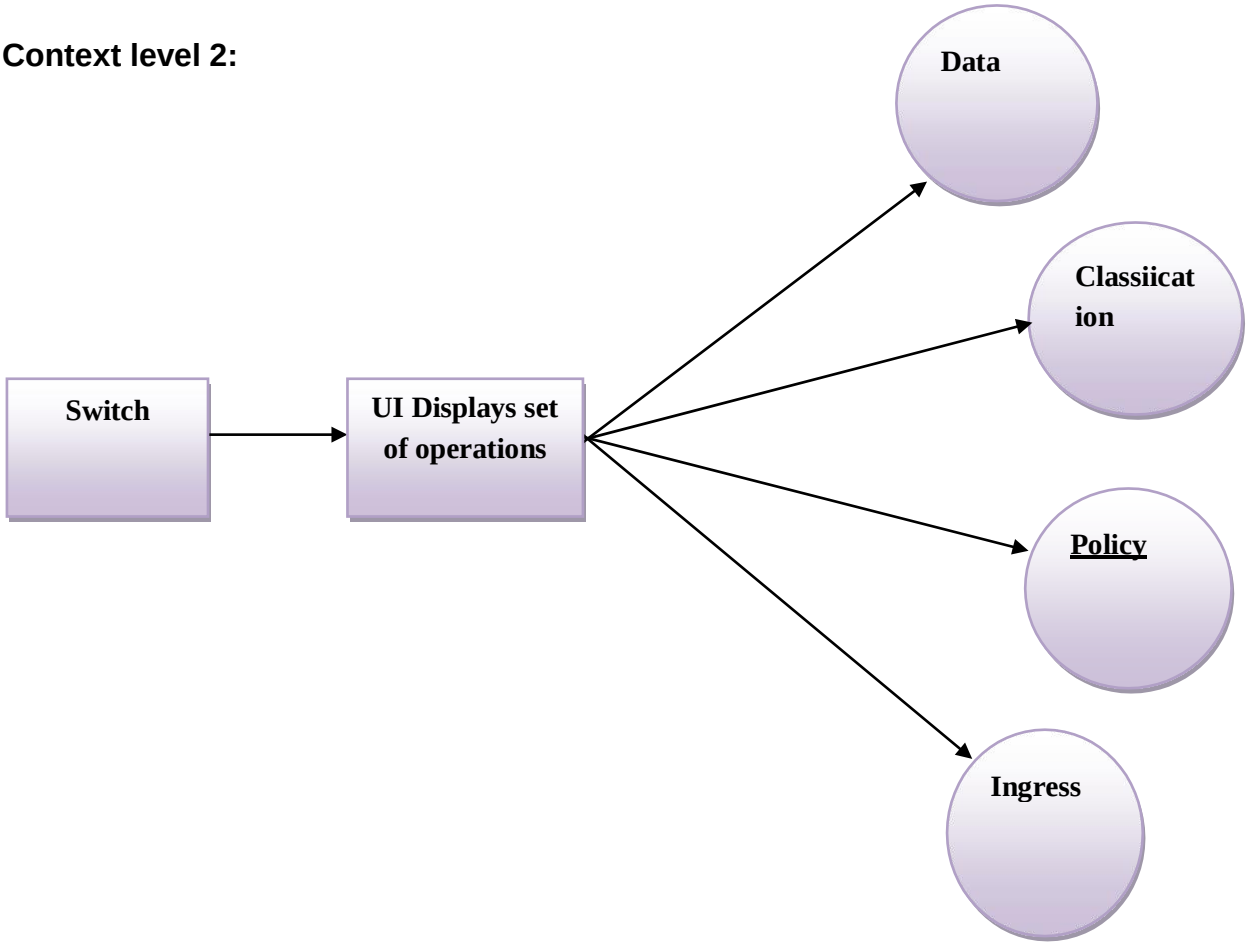
A New Architecture for Network Intrusion Detection and Prevention

System Process

Login DFD



Context level 2:



UML Diagrams:

What is a UML Use Case Diagram?

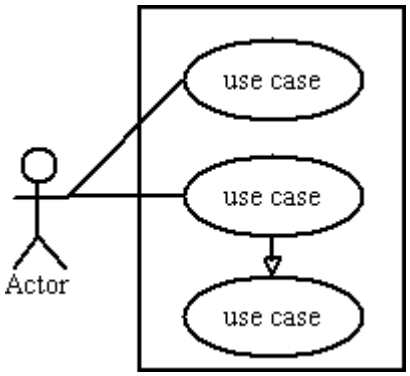
Use case diagrams model the functionality of a system using actors and use cases. Usecases are services or functions provided by the system to its users.

Basic Use Case Diagram Symbols and Notations

System

Draw your system's boundaries using a rectangle that contains use cases. Place actorsoutside the system's boundaries.

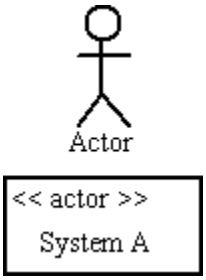
Use Case



Draw use cases using ovals. Label with ovals with verbs that represent the system'sfunctions.

Actors

Actors are the users of a system. When one system is the actor of another system, label theactor system with the actor stereotype.



Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.

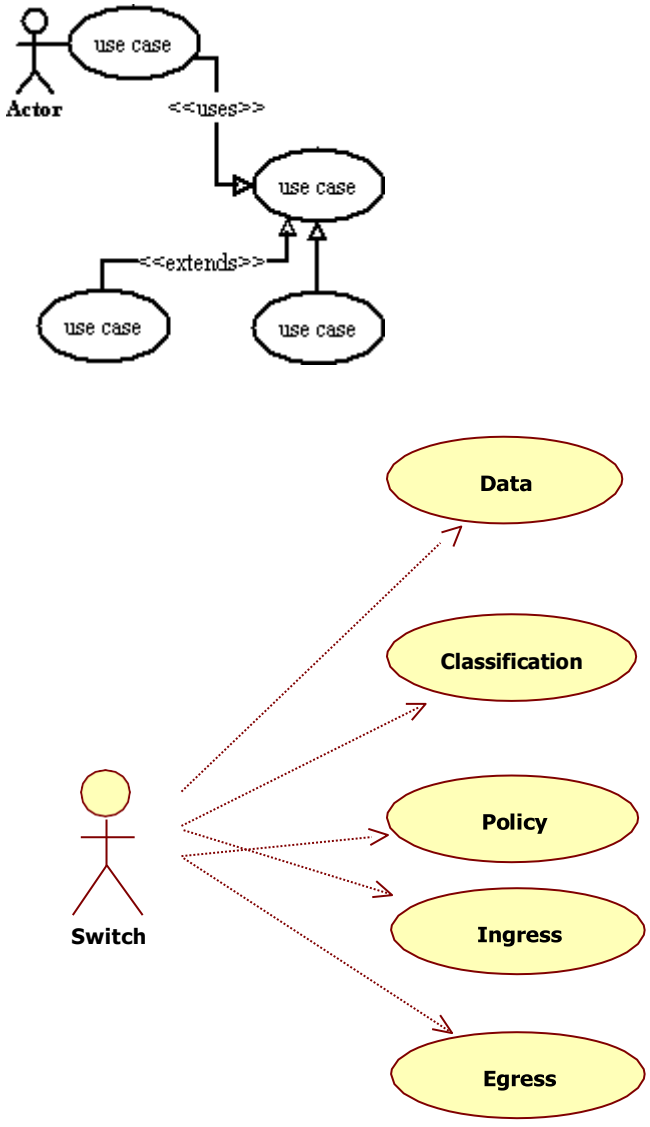


Fig: Use Case diagram

Activity Diagram

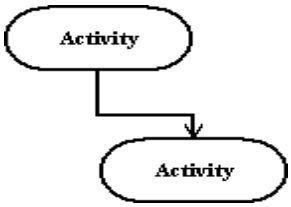
An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

Basic Activity Diagram Symbols and NotationsAction states

Action states represent the non interruptible actions of objects. You can draw an action statein Smart Draw using a rectangle with rounded corners.

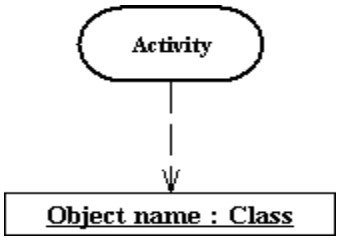
Action Flow

Action flow arrows illustrate the relationships among action states.



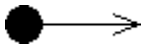
Object Flow

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



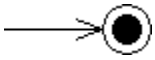
Initial State

A filled circle followed by an arrow represents the initial action state.



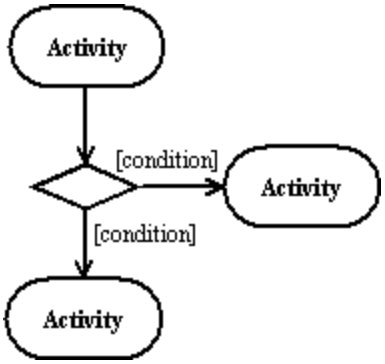
Final State

An arrow pointing to a filled circle nested inside another circle represents the final actionstate.



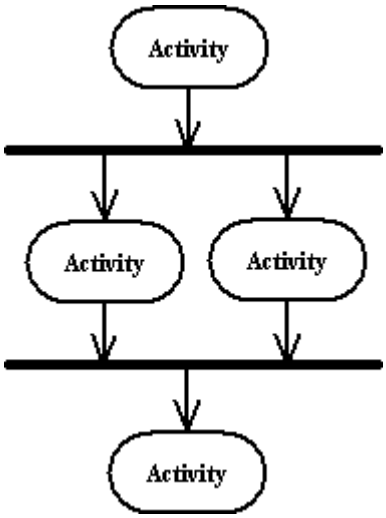
Branching

A diamond represents a decision with alternate paths. The outgoing alternates should belabeled with a condition or guard expression. You can also label one of the paths "else."



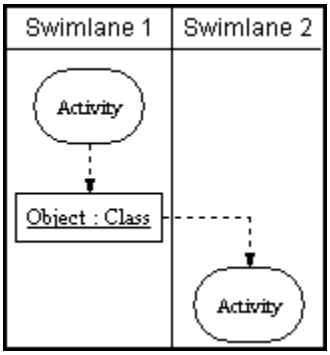
Synchronization

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



Swim Lane

Swim lanes group related activities into one column.



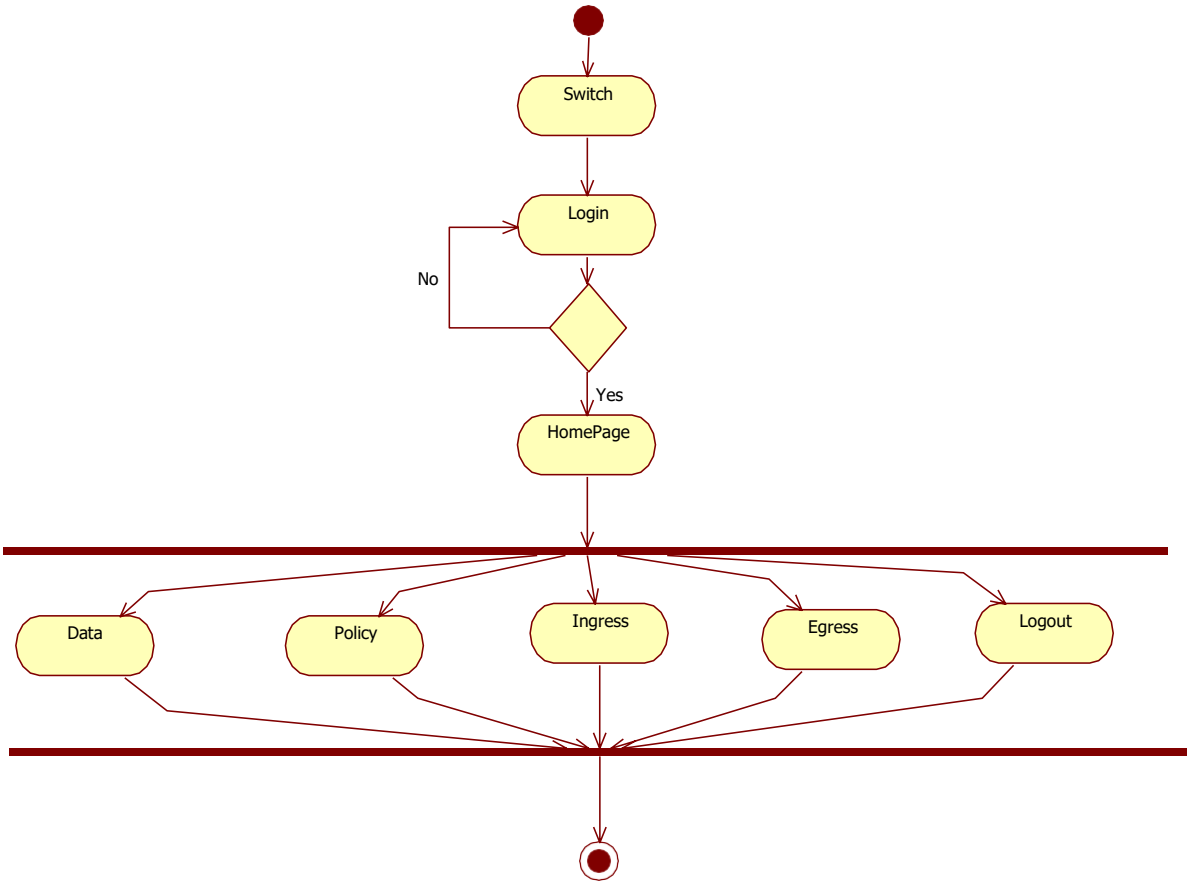


Fig: Activity diagram

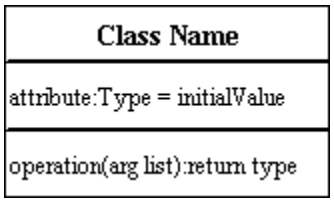
Class Diagram

Class diagrams are the backbone of almost every object-oriented method including UML.They describe the static structure of a system.

BASIC CLASS DIAGRAM SYMBOLS AND NOTATIONS:

Classes represent an abstraction of entities with common characteristics. Associationsrepresent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and Write operations into the third.

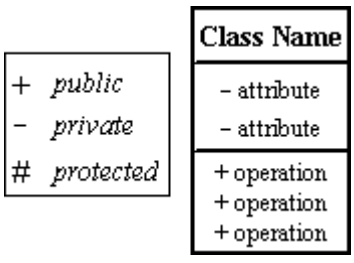


Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

Visibility

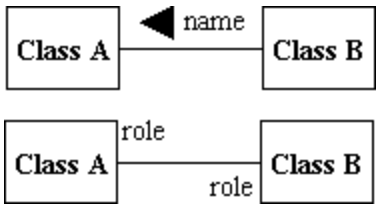
Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.



ASSOCIATIONS

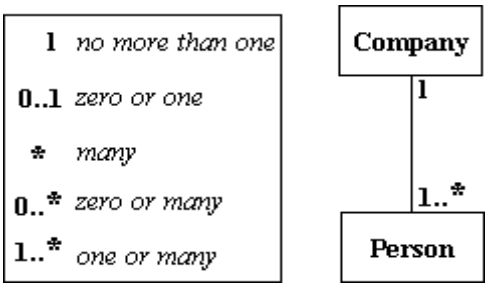
Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

Note*:* It's uncommon to name both the association and the class roles.



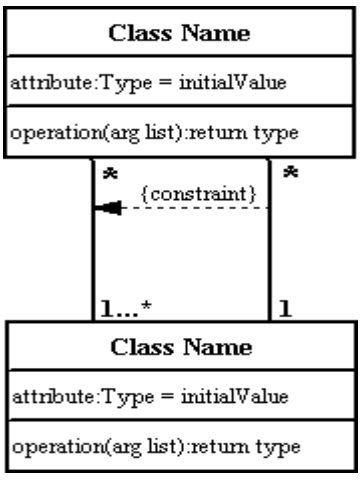
Multiplicity (Cardinality)

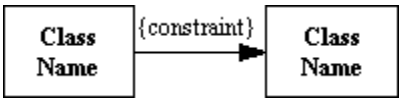
Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, onecompany will have one or more employees, but each employee works for one companyonly.



Constraint

Place constraints inside curly braces {}.

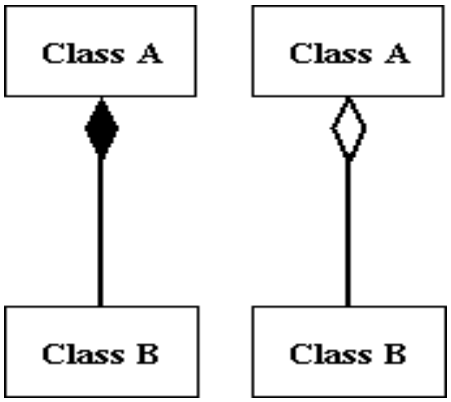




Simple Constraint

Composition and Aggregation

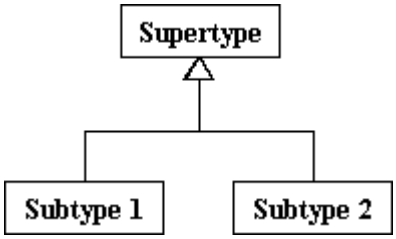
Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate **composition** with a filled diamond. Use a hollow diamond to represent a simple **aggregation** relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



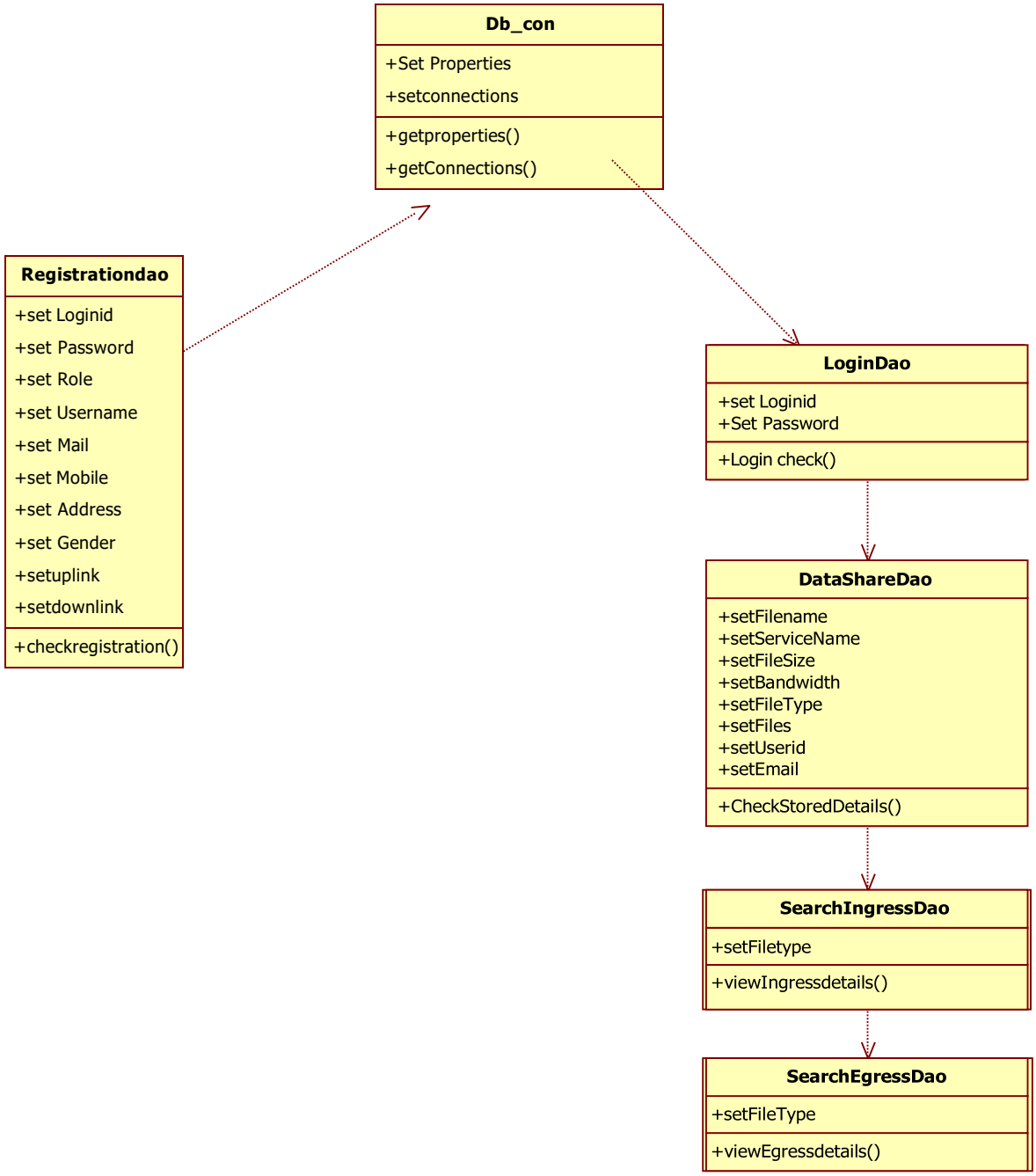
Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to arelationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.

In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other



hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.


Db_con
+Set Properties
+setconnections
+getproperties()
+getConnections()
Registrationdao
+set Loginid
+set Password
+set Role
+set Username
+set Mail
+set Mobile
+set Address
+set Gender
+setuplink
+setdownlink
+checkregistration()
LoginDao
+set Loginid
+Set Password
+Login check()
DataShareDao
+setFilename
+setServiceName
+setFileSize
+setBandwidth
+setFileType
+setFiles
+setUserid
+setEmail
+CheckStoredDetails()
SearchIngressDao
+setFiletype
+viewIngressdetails()
SearchEgressDao
+setFileType
+viewEgressdetails()

State chart Diagram

A state chart diagram shows the behavior of classes in response to external stimuli. Thisdiagram models the dynamic flow of control from state to state within a system.

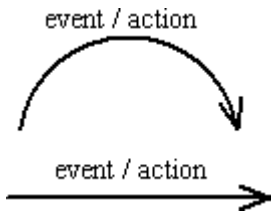
Basic State chart Diagram Symbols and Notations

States

States represent situations during the life of an object. You can easily illustrate a state inSmart Draw by using a rectangle with rounded corners.

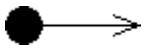
Transition

A solid arrow represents the path between different states of an object. Label the transitionwith the event that triggered it and the action that results from it.



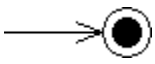
Initial State

A filled circle followed by an arrow represents the object's initial state.



Final State

An arrow pointing to a filled circle nested inside another circle represents the object's finalstate.



Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.

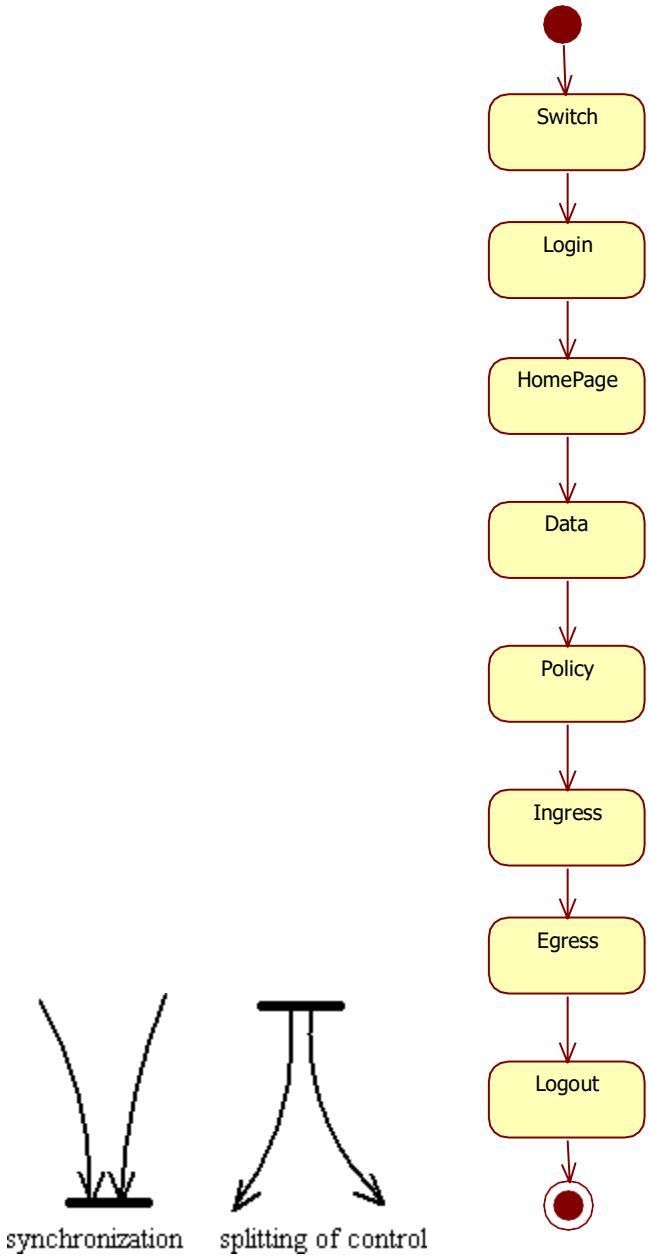


Fig: State Chart diagram

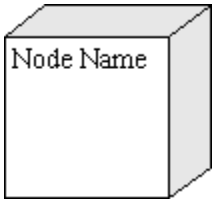
Deployment Diagram

Deployment diagrams depict the physical resources in a system including nodes,components, and connections.

Basic Deployment Diagram Symbols and Notations

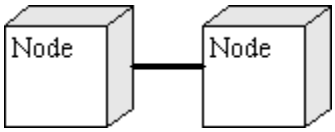
Component

A node is a physical resource that executes code components. Learn how to resize grouped objects like nodes.



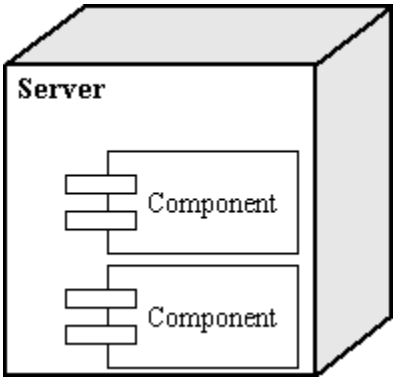
Association

Association refers to a physical connection between nodes, such as Ethernet. Learn how to connect two nodes.

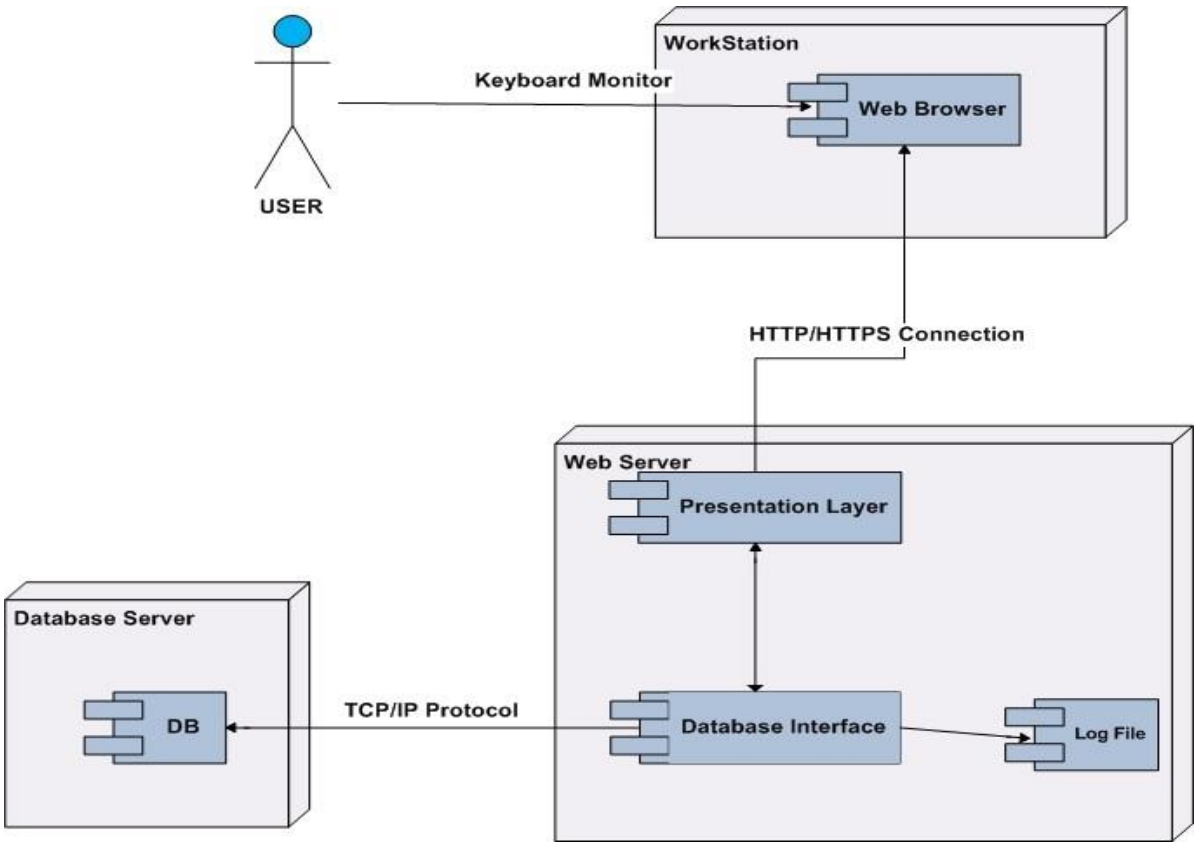


Components and Nodes

Place components inside the node that deploys them.



DEPLOYMENT DIAGRAM:



Component Diagram

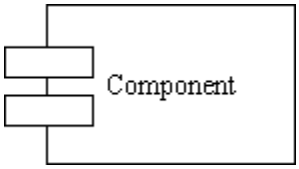
A component diagram describes the organization of the physical components in a system.

Basic Component Diagram Symbols and Notations

Component

A component is a physical building block of the system. It is represented as a rectangle withtabs.

Learn how to resize grouped objects like components.

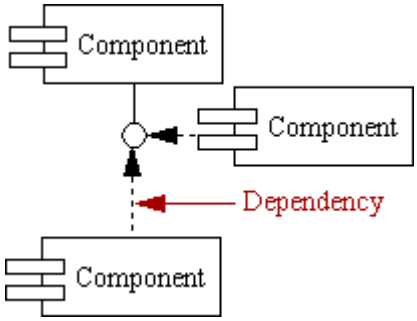


Interface

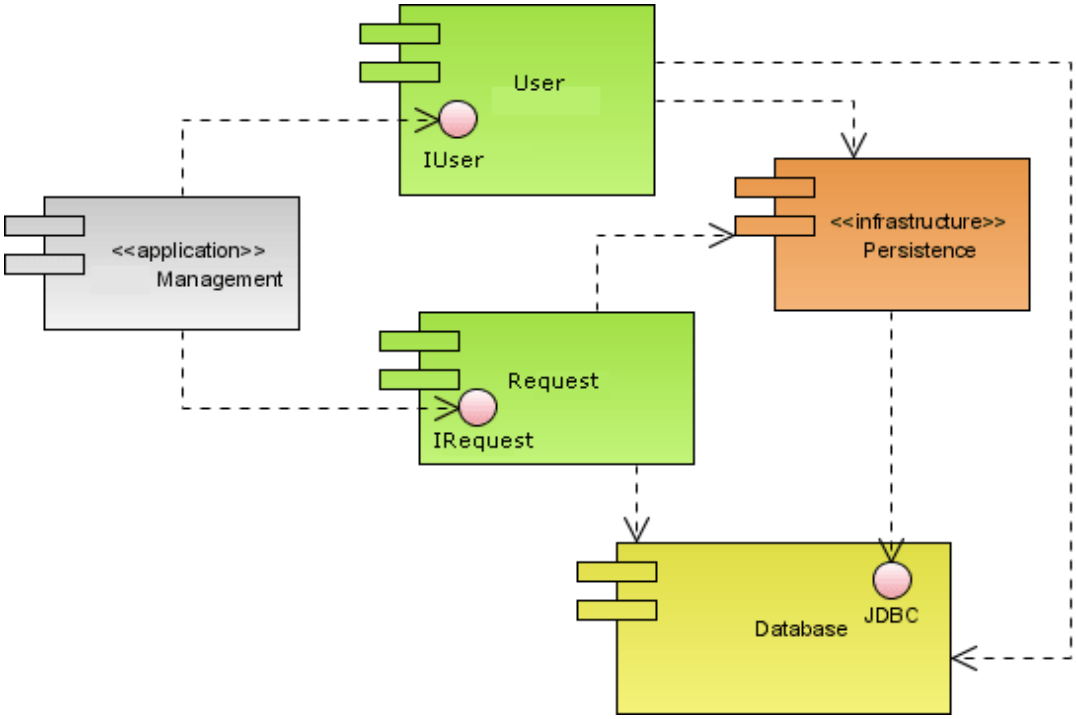
An interface describes a group of operations used or created by components.

Dependencies

Draw dependencies among components using dashed arrows. Learn about line styles in SmartDraw.



COMPONENT DIAGRAM:



E-R DIAGRAM:

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams(DFDs), which map out the flow of information for processes or systems.

Column Name	Data Type	Nullable	Default	Primary Key
SID	NUMBER	No	-	1
SERVICENAME	VARCHAR2(4000)	Yes	-	-
STORAGE	NUMBER	Yes	-	-
MEMORYRESERVATION	VARCHAR2(4000)	Yes	-	-
PRIORITY	VARCHAR2(4000)	Yes	-	-
STATUS	VARCHAR2(4000)	Yes	-	-
				1 - 6

6. IMPLEMENTATION

Vehicle Detection

Given the fact that the 2018 AI City Challenge dataset did not provide any ground-truth detection or tracking annotations, we were not able to train a vehicle detection model specific to this dataset. Instead, we rely on transfer learning, taking advantage of state-of-the-art deep learning models that have been previously trained for this task. Specifically, the videos in Tracks 1 and 3 of the dataset are similar in quality and scope to videos used for the object detection, localization, and classification task of the 2017 AI City Challenge [14]. As such, we have chosen to rely on the top-2 best performing models from that challenge, the 3D Deformable model by Tang et al. [16] and our lab's submission to that challenge, the model by Bhandary et al. [2]. Both models provide as output, for each frame, a series of bounding-boxes believed to contain an object of interest, the class of that object, and a confidence score. The 2017 challenge sought to localize and classify objects in 14 categories, including car, suv, smalltruck, mediumtruck, largetruck, pedestrian, bus, van, groupofpeople, bicycle, motorcycle, trafficsignal-green, trafficsignalyellow, and trafficsignal-red. In order to maximize the utility of the detections we will provide as input to our algorithm, we filter the detector output as follows: • Remove detections with a confidence less than some threshold α. • Remove detections for non-vehicle classes, keeping only those for the car, suv, smalltruck, mediumtruck, largetruck, bus, van, bicycle, and motorcycle classes. • Filter bounding boxes within the same frame that have an overlap, measured by the Intersection-over-Union (IOU) score, of at least some threshold β. Detections are filtered while traversing the frame in a left-right top-down order when the IOU of the detection with an already selected bounding box is greater than β.

Vehicle Tracking

Our vehicle tracking algorithm relies on localization results from the vehicle detection methods described in Section 4.1, which are enhanced with optical-flow based features to provide robust vehicle trajectories.

Tracking-by-Detection

Given the fact that the input footage was at 1080p resolution and 30 fps, objects move few pixels from one frame

Speed Estimation

Our method takes a data-driven approach to estimating the speed of vehicles and relies on several strong assumptions. First, the camera recording traffic should be static, which holds for the 2018 AI City Challenge. Secondly, we assume that the maximum speed limit is known for the road segments captured in the footage and at least one vehicle drives on the segment at that speed

7. TECHNOLOGY DESCRIPTION

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python
print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not
assert finally or
break for pass
class from print
continue global raise
def if return
del import try
elif in while
else is with
except lambda yield

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python
import sys

try:
   # open file stream
```

```
  file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

."

```
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
sentence = "This is a sentence
```

Live Demo

```
#!/usr/bin/python

# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline is a sample snip using the semicolon.
```

import sys; x = 'foo'; sys.stdout.write(x + '\n')

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

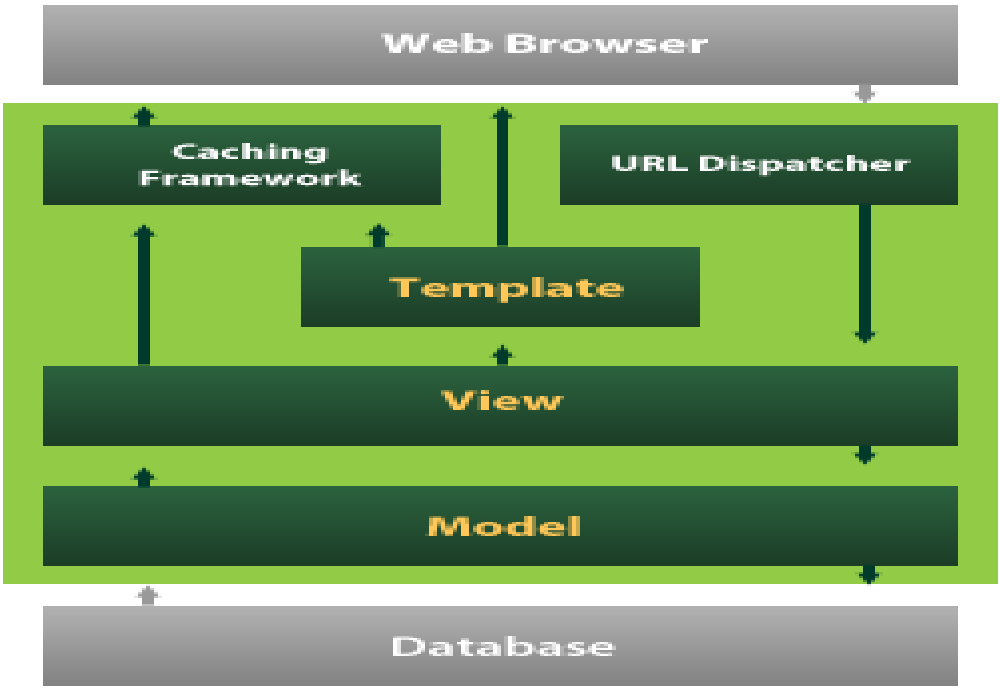
Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

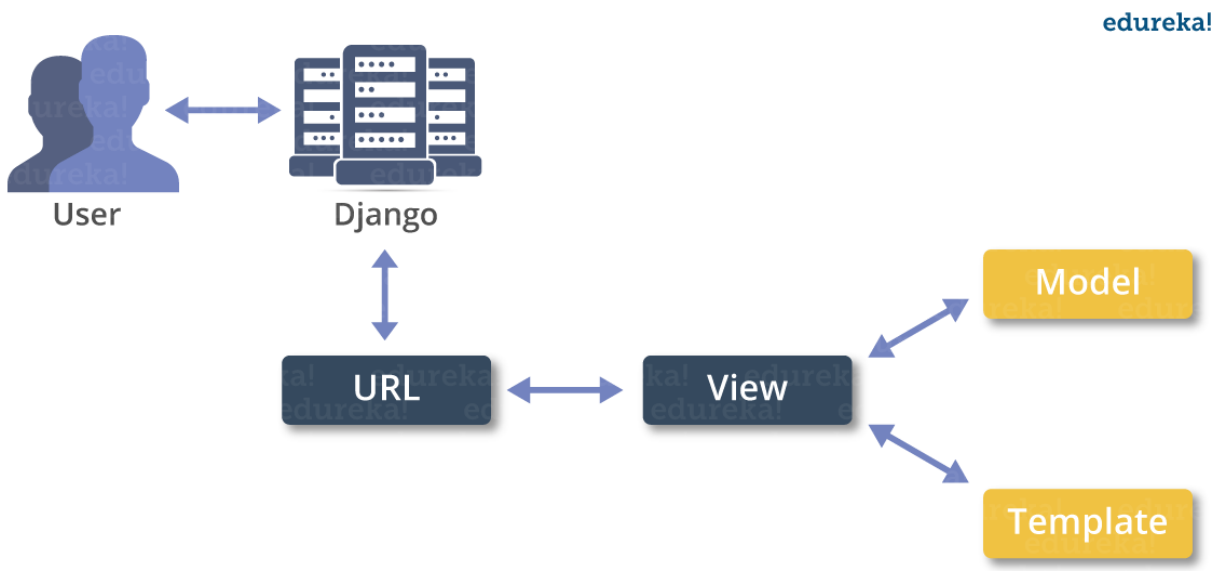
DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

myproject/

manage.py

myproject/

__init__.py

settings.py

urls.py

wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files

__init__.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development model

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
   'PASSWORD': '',
    'HOST': '',
    'PORT': '',
  }
}
```

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostGreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...
0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
```

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py –

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

myapp/

```
  __init__.py
  admin.py
  models.py
  tests.py
  views.py
```

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
  'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms
class LoginForm(forms.Form):
  user = forms.CharField(max_length = 100)
  password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm
def login(request):
  username = "not logged in"

  if request.method == "POST":
    #Get the posted form
    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
  else:
    MyLoginForm = Loginform()
  return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
  <html>
```

```
  <body>
  <form name = "form" action = "{% url "myapp.views.login" %}"
     method = "POST" >{% csrf_token %}
   <div style = "max-width:470px;">
    <center>

 <input type = "text" style = "margin-left:20%;"
       placeholder = "Identifiant" name = "username" />
       </center>
       </div>
       <br>
       <div style = "max-width:470px;">
          <center>
        <input type = "password" style = "margin-left:20%;"
         placeholder = "password" name = "password" />
        </center>
       </div>
        <br>
       <div style = "max-width:470px;">
        <center>
            <button style = "border:0px; background-color:#4285F4; margin-top:8%;
              height:35px; width:80%;margin-left:19%;" type = "submit"
              value = "Login" >
            <strong>Login</strong>
            </button>   </center>
 <div>
 </form>
</body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>  <body>
```

```
  You are : <strong>{{username}}</strong>
  </body>
</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url

from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
  url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −

'django.contrib.sessions.middleware.SessionMiddleware'

And INSTALLED_APPS should have −

'django.contrib.sessions'

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

```
def login(request):
  username = 'not logged in'
  if request.method == 'POST':
    MyLoginForm = LoginForm(request.POST)
```

```
   if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
      request.session['username'] = username
   else:
      MyLoginForm = LoginForm()
   return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
   if request.session.has_key('username'):
      username = request.session['username']
      return render(request, 'loggedin.html', {"username" : username})
   else:
      return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',
   url(r'^connection/','formView', name = 'loginform'),
   url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

7. CODING

Source Code

DJANGO MAIN METHOD USING PYTHON:

```
#!/usr/bin/env python
importos
import sys

if __name__ == "__main__":
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "{{ project_name }}.settings")
try:
fromdjango.core.management import execute_from_command_line
exceptImportError:
    # The above import may fail for some other reason. Ensure that the
    # issue is really that Django is missing to avoid masking other
    # exceptions on Python 2.
try:
importdjango
exceptImportError:
raiseImportError(
"Couldn't import Django. Are you sure it's installed and "
        "available on your PYTHONPATH environment variable? Did you "
        "forget to activate a virtual environment?"
      )
raise
execute_from_command_line(sys.argv)
```

VIEW'sSOURCECODE:

```
importos

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
#See https://docs.djangoproject.com/en/{{ docs_version }}/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '{{ secret_key }}'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = '{{ project_name }}.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
```

```
        'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = '{{ project_name }}.wsgi.application'
# Database
# https://docs.djangoproject.com/en/{{ docs_version }}/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
# Password validation
# https://docs.djangoproject.com/en/{{ docs_version }}/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/{{ docs_version }}/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/{{ docs_version }}/howto/static-files/

STATIC_URL = '/static/'
```

URLsSOURCE CODE:

```
"""{{ project_name }} URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/{{ docs_version }}/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')



Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
fromdjango.conf.urls import url
fromdjango.contrib import admin
urlpatterns = [
url(r'^admin/', admin.site.urls),
```

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9. SAMPLE CODE

5.1 SampleCode

5.1.1 XML Code page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/ap
k/res/android" android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<android.support.v7.widget.LinearLayou
tCompat
android:layout_width="match_parent"
android:layout_height="150dp"
android:background="@color/colorText
Hint" android:gravity="center"
android:orientation="vertical">

<android.support.v7.widget.AppCompat
TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Bookings"
android:textSize="20sp" />
```

```
<android.support.v7.widget.AppCompat
TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:text="@string/text_hello" />
<android.support.v7.widget.AppCompat
TextView
android:id="@+id/textViewName"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</android.support.v7.widget.LinearLayoutCompat>

<android.support.v7.widget.AppCompat
TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:paddingBottom="5dp"
android:paddingLeft="16dp"
android:paddingTop="5dp"
android:text="Bookings"
android:textColor="@android:color/bla
ck" />

<ScrollView
android:id="@+id/scrollview"
android:layout_width="fill_paren
t"
android:layout_height="fill_paren
t"
android:layout_below="@+id/spin
ner1"
android:layout_alignParentBotto
m="true"
```

```
    android:layout_alignParentLeft="
    true">
    <LinearLayout
    android:layout_width="match_
    parent"
    android:layout_height="562dp"

    android:layout_marginBottom=
    "200dp"
    android:orientation="vertical"
    >

    <listView



android:id="@+id/listView1"
android:layout_width="wrap_con
tent"
android:layout_height="600dp"
android:layout_marginLeft="0dp
" >
    </ListView>

    </LinearLayout>
    </ScrollView>

    </LinearLayout>
```

5.1.2 Java Code page

```
package com.example.spaceimpactor.houser.activities;

import
java.util.ArrayList; import
android.os.Bundle; import
android.app.Activity;
import
android.content.Context;
import
android.content.Intent;
importandroid.database.C
ursor;
import android.database.sqlite.SQLiteDatabase;
import android.view.View;
import
android.view.View.OnClickListene
r; import
android.widget.AdapterView;
import

android.widget.ArrayAdapter;
import android.widget.Button;
import
android.widget.EditText;
import
android.widget.ImageView;
import
android.widget.ListView;
import
android.widget.Spinner;
import
android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Toast;
```

```
import com.example.spaceimpactor.houser.R;
public class Bookings extends
   Activity{ Spinner sp;
ImageViewout;
TextViewaaa;
SQLiteDatabase
db; ListViewl;
EditTextt1;
ArrayList<String>
list1;
ArrayAdapterada
pter;
   Button sub;
   String lmb,lser,lem;
@Override
protected void onCreate(Bundle
savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_bookings);

final
GlobalClassglobalvariabel=(GlobalClass)getApplicationContext
(); aaa=(TextView)findViewById(R.id.textViewName);
aaa.setText(globalvariabel.GetUsername().toString());

db=openOrCreateDatabase("ServiceProvider",
Context.MODE_PRIVATE, null); l = (ListView)
findViewById(R.id.listView1);
final ArrayList<String> list = new ArrayList<String>();
list1 = new ArrayList<String>();
      Cursor res = db.rawQuery("SELECT * FROM book where uid='" +
aaa.getText() + "'", null); if (res.getCount() != 0) {
while (res.moveToNext()) {
list.add("Name: " + res.getString(1) + "\nMobile No: " + res.getString(2) + "\n" + "Service
Type: " + res.getString(3) + "\n" + "Email: " + res.getString(4)+ "\n" + "Booking Date: "
+ res.getString(5));
```

```
list1.add(res.getString(1));
        }
      }
adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, list);
l.setAdapter(adapter);
   }

}
```

5.1.3 Java Code

```
package com.example.spaceimpactor.houser.fragment;

import android.content.Context;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import
android.support.annotation.NonNull
; import
android.support.v4.app.Fragment;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import android.support.annotation.Nullable;
import android.support.design.widget.Snackbar;
import
android.support.design.widget.TextInputEditTe
xt; import
android.support.design.widget.TextInputLayout
; import
android.support.v4.widget.NestedScrollView;
import
```

```
android.support.v7.app.AppCompatActivity;
import
android.support.v7.widget.AppCompatButton;
import
android.support.v7.widget.AppCompatTextVie
w; import android.widget.Button;
import
android.widget.EditText;
import
android.widget.ProgressBar;
import
android.widget.TextView;
import android.widget.Toast;

import
com.google.firebase.auth.FirebaseAuth;
import
com.google.firebase.auth.FirebaseUser;
import
com.google.firebase.database.DataSnapshot;
import
com.google.firebase.database.DatabaseError
;
import
com.google.firebase.database.DatabaseReferenc
e; import
com.google.firebase.database.FirebaseDatabase;
import
com.google.firebase.database.ValueEventListene
r;

public class ProfileFragmentextends Fragment implements View.OnClickListener {

private NestedScrollViewnestedScrollView;
```

```
private
TextInputLayouttextInputLayoutName;

private
TextInputLayouttextInputLayoutPhone;
private
TextInputLayouttextInputLayoutEmail;
private
TextInputLayouttextInputLayoutPasswor
d;
private TextInputLayouttextInputLayoutConfirmPassword;

private
TextInputEditTexttextInputEditTextName;
private
TextInputEditTexttextInputEditTextPhone;
private
TextInputEditTexttextInputEditTextEmail;
private
TextInputEditTexttextInputEditTextPasswo
rd;
private TextInputEditTexttextInputEditTextConfirmPassword;

private AppCompatButtonappCompatButtonRegister;

private
InputValidationinputValidation;
private
DatabaseHelperdatabaseHelper;
private User user;
```

```
EditTextid,ps,em,
   mb; Button
   sub;
SQLiteDatabase
db;
TextViewaaa;

private static final String TAG = MainActivity.class.getSimpleName();
private TextViewtxtDetails;
private EditTextinputName,
inputEmail; private Button
btnSave;
private DatabaseReferencemFirebaseDatabase;
private FirebaseDatabasemFirebaseInstance;

private String userId;

private Button btnChangeEmail, btnChangePassword, btnSendResetEmail,
btnRemoveUser, changeEmail, changePassword, sendEmail, remove,
signOut;

private EditTextoldEmail, newEmail, password,
newPassword; private ProgressBarprogressBar;
private FirebaseAuth.AuthStateListenerauthListener;
private FirebaseAuthauth;


@Override
public View onCreateView(LayoutInflaterinflater,
                ViewGroup container, Bundle
                savedInstanceState) {
// Inflate the layout for this fragment
View myView = inflater.inflate(R.layout.fragment_profile, container, false);
//     appCompatButtonRegister =
(AppCompatButton)myView.findViewById(R.id.appCompatButtonRegister);
//     appCompatButtonRegister.setOnClickListener(this);
```

```
return myView;

   }
private void createUser(String name, String email) {
// TODO
// In real apps this userId should be
                    fetched
  // by implementing firebase auth
if (TextUtils.isEmpty(userId)) {
userId= mFirebaseDatabase.push().getKey();
      }

      User user = new User();
mFirebaseDatabase.child(userId).setVal
ue(user); addUserChangeListener();
   }

private void addUserChangeListener() {
// User data change listener
mFirebaseDatabase.child(userId).addValueEventListener(new
ValueEventListener() { @Override
public void
            onDataChange(DataSnapshotdataSnaps
            hot) { User user =
            dataSnapshot.getValue(User.class);

// Check for null
if (user == null) {
Log.e(TAG, "User data is
null!"); return;
            }
inputEmail.setText
("");
inputName.setText
("");

         }
```

```
@Override
public void onCancelled(DatabaseError error) {
// Failed to read value
Log.e(TAG, "Failed to read user", error.toException());
        }
    });
  }

private void updateUser(String name, String email) {
// updating the user via child nodes
if (!TextUtils.isEmpty(name))
mFirebaseDatabase.child(userId).child("name").setValue(name);

if (!TextUtils.isEmpty(email))
mFirebaseDatabase.child(userId).child("email").setValue(email);
  }

//sign out method
public void
signOut() {
auth.signOut();
  }

@Override
public void onResume() {
super.onResume();
progressBar.setVisibility(View.
GONE);
  }

@Override
public void onStart() {
super.onStart();
auth.addAuthStateListener(authLi
stener);
  }
```

```
@Override
public void onStop() {
super.onStop();
if (authListener!= null) {
auth.removeAuthStateListener(authListener);
    }
  }

public void onViewCreated(@NonNullView view, @NullableBundle savedInstanceState) {
super.onViewCreated(view, savedInstanceState);
//     initViews();
//     initObjects();
auth= FirebaseAuth.getInstance();

//get current user
final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

authListener= new
FirebaseAuth.AuthStateListener() {
@Override
public void
onAuthStateChanged(@NonNullFirebaseAuthfirebaseAuth)
{ FirebaseUser user = firebaseAuth.getCurrentUser();
if (user == null) {
// user auth state is changed - user is null
            // launch login activity
startActivity(new Intent(getActivity(), LoginActivity.class));
          }
        }
    };
```

```
//      btnChangeEmail = (Button)findViewById(R.id.change_email_button);
      //btnChangePassword = (Button)
findViewById(R.id.change_password_button); btnSendResetEmail=
(Button) getView().findViewById(R.id.sending_pass_reset_button);
btnRemoveUser= (Button)
getView().findViewById(R.id.remove_user_button);
//      changeEmail = (Button)getView().findViewById(R.id.changeEmail);
//      chanPtView().findViewById(R.id.changePass);


sendEmail=                          (Button)
getView().findViewById(R.id.send);  remove  =
(Button)
getView().findViewById(R.id.remove);
signOut=                            (Button)
getView().findViewById(R.id.sign_out);

oldEmail= (EditText)
getView().findViewById(R.id.old_email); newEmail=
(EditText) getView().findViewById(R.id.new_email);
password = (EditText)
getView().findViewById(R.id.password); newPassword=
(EditText) getView().findViewById(R.id.newPassword);

oldEmail.setVisibility(View.GO
NE);
newEmail.setVisibility(View.GO
NE);
password.setVisibility(View.GO
NE);
newPassword.setVisibility(View.
GONE);
//   changeEmail.setVisibility(View.GONE);
//
       changePassword.setVisibility(View.
GONE);
```

```
sendEmail.setVisibility(View.GONE);
remove.setVisibility(View.GONE);

progressBar= (ProgressBar) getView().findViewById(R.id.progressBar);

if (progressBar!= null) {
progressBar.setVisibility(View.GONE);
    }
btnSendResetEmail.setOnClickListener(new
View.OnClickListener() { @Override
public void onClick(View v) {
oldEmail.setVisibility(View.VIS
IBLE);
newEmail.setVisibility(View.GO
NE);
password.setVisibility(View.GO
NE);
newPassword.setVisibility(View.
GONE);
//          changeEmail.setVisibility(View.GONE);
          //
               changePassword.setVisibility(View.
GONE);
sendEmail.setVisibility(View.VISIBLE);
remove.setVisibility(View.GONE);
       }
    });
sendEmail.setOnClickListener(new
View.OnClickListener() { @Override
public void onClick(View v) {
progressBar.setVisibility(View.VISIBLE);
if (!oldEmail.getText().toString().trim().equals("")) {
auth.sendPasswordResetEmail(oldEmail.getText().toString().trim())
                .addOnCompleteListener(new OnCompleteListener<Void>() {
```

```
@Override
public void onComplete(@NonNullTask<Void> task) {
if (task.isSuccessful()) {
Toast.makeText(getActivity(), "Reset password email is sent!", Toast.LENGTH_SHORT).show();
progressBar.setVisibility(View.GONE);
                    } else {
Toast.makeText(getActivity(), "Failed to send reset email!", Toast.LENGTH_SHORT).show();
progressBar.setVisibility(View.GONE);
                    }
                }
            });
        } else {
oldEmail.setError("Enter
email");
progressBar.setVisibility(View.
GONE);
        }
    }
});
btnRemoveUser.setOnClickListener(new
View.OnClickListener() { @Override
public void onClick(View v) {
progressBar.setVisibility(View.VISIBLE);
if (user !=
null) {
user.delete()   .addOnCompleteListener(new OnCompleteListener<Void>() {

@Override
```

```
public void onComplete(@NonNullTask<Void> task) {
if (task.isSuccessful()) {
Toast.makeText(getActivity(), "Your profile is deleted:( Create a account now!",
Toast.LENGTH_SHORT).show(); startActivity(new Intent(getActivity(),
RegisterActivity.class));
//                          finish();
progressBar.setVisibility(View.GONE);
                        } else {
Toast.makeText(getActivity(), "Failed to delete your account!", Toast.LENGTH_SHORT).show();
progressBar.setVisibility(View.GONE);
                        }
                    }
                });
            }
        }
    });


signOut.setOnClickListener(new
View.OnClickListener() { @Override
public void
onClick(View v) {
signOut();
          }
      });
//txtDetails = (TextView)
findViewById(R.id.txt_user); inputName=
(EditText) getView().findViewById(R.id.name);
inputEmail= (EditText)
getView().findViewById(R.id.email); btnSave=
(Button)
getView().findViewById(R.id.btn_save);


mFirebaseInstance= FirebaseDatabase.getInstance();


// get reference to 'users' node
```

```
mFirebaseDatabase= mFirebaseInstance.getReference("users");

// store app title to 'app_title' node
mFirebaseInstance.getReference("app_title").setValue("Houser");

// app_title change listener
mFirebaseInstance.getReference("app_title").addValueEventListener(new
ValueEventListener() { @Override
public void
onDataChange(DataSnapshotdataSnapshot) {
Log.e(TAG, "App title updated");

        String appTitle = dataSnapshot.getValue(String.class);

// update toolbar title
        //getSupportActionBar().setTitle(appTitle);
}

@Override
public void onCancelled(DatabaseError error) {
// Failed to read value
Log.e(TAG, "Failed to read app title value.", error.toException());
      }
    });

// Save / update the user
btnSave.setOnClickListener(new
View.OnClickListener() { @Override
public void onClick(View view) {
        String name =
        inputName.getText().toString();
        String email =
        inputEmail.getText().toString();
```

```
// Check for already existed
userId if
(TextUtils.isEmpty(userId))
{ createUser(name, email);
            } else {

updateUser(name, email);
            }
        }
    });


  }

@Override
public void
onClick(View v) {
postDataToSQLite();
  }

private void initViews() {
nestedScrollView= (NestedScrollView) getView().findViewById(R.id.nestedScrollView);

textInputLayoutName= (TextInputLayout)
getView().findViewById(R.id.textInputLayoutName); textInputLayoutPhone=
(TextInputLayout) getView().findViewById(R.id.textInputLayoutPhone);
textInputLayoutEmail= (TextInputLayout)
getView().findViewById(R.id.textInputLayoutEmail); textInputLayoutPassword=
(TextInputLayout) getView().findViewById(R.id.textInputLayoutPassword);
textInputLayoutConfirmPassword= (TextInputLayout)
getView().findViewById(R.id.textInputLayoutConfirmPassword);

textInputEditTextName= (TextInputEditText)
getView().findViewById(R.id.textInputEditTextName); textInputEditTextPhone=
(TextInputEditText) getView().findViewById(R.id.textInputEditTextPhone);
textInputEditTextEmail= (TextInputEditText)
```

```
getView().findViewById(R.id.textInputEditTextEmail); textInputEditTextPassword=
(TextInputEditText) getView().findViewById(R.id.textInputEditTextPassword);
textInputEditTextConfirmPassword= (TextInputEditText)
getView().findViewById(R.id.textInputEditTextConfirmPassword);
   }

private void initObjects() {
inputValidation= new
InputValidation(getActivity());
databaseHelper= new
DatabaseHelper(getActivity()); user = new
User();

   }
private void postDataToSQLite() {
if (!inputValidation.isInputEditTextFilled(textInputEditTextName,
textInputLayoutName, getString(R.string.error_message_name))) {
return;
     }
if (!inputValidation.isInputEditTextFilled(textInputEditTextPhone,
textInputLayoutPhone, "Enter Phone Number")) {
return;
     }
if (!inputValidation.isInputEditTextFilled(textInputEditTextEmail,
textInputLayoutEmail, getString(R.string.error_message_email))) {
return;
     }
if (!inputValidation.isInputEditTextFilled(textInputEditTextEmail,
textInputLayoutEmail, getString(R.string.error_message_email))) {
return;
     }
if (!inputValidation.isInputEditTextEmail(textInputEditTextEmail,
textInputLayoutEmail, getString(R.string.error_message_email))) {
return;
```

```
        }
    if (!inputValidation.isInputEditTextPhone(textInputEditTextPhone,
    textInputLayoutPhone, getString(R.string.error_message_phone))) {
    return;
        }
    if (!inputValidation.isInputEditTextFilled(textInputEditTextPassword,
    textInputLayoutPassword,
    getString(R.string.error_message_password))) {
    return;
        }
    if (!inputValidation.isInputEditTextMatches(textInputEditTextPassword,
    textInputEditTextConfirmPassword, textInputLayoutConfirmPassword,
    getString(R.string.error_password_match))) {
    return;
        }

    if (!databaseHelper.checkUser(textInputEditTextEmail.getText().toString().trim())) {

    user.setName(textInputEditTextName.getText().toString().t
    rim());
    user.setPhone(textInputEditTextPhone.getText().toString().
    trim());
    user.setEmail(textInputEditTextEmail.getText().toString().t
    rim());
    user.setPassword(textInputEditTextPassword.getText().toS
    tring().trim());
    databaseHelper.updateUser(
    user); emptyInputEditText();
        }
     }
    private void emptyInputEditText()
     {
```

```
textInputEditTextName.setText(null);
textInputEditTextPhone.setText(null);
textInputEditTextEmail.setText(null);
textInputEditTextPassword.setText(null
);
textInputEditTextConfirmPassword.set
Text(null);
   }

}
```

5.1.3 XML Code

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.v4.widget.NestedScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/
res-auto"
android:id="@+id/nestedScrollView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="20dp"
android:paddingLeft="20dp"
android:paddingRight="20dp"
android:paddingTop="20dp">

<LinearLayout
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_m
argin"
android:paddingLeft="@dimen/activity_horizontal_m
argin"
android:paddingRight="@dimen/activity_horizontal_
margin" android:paddingTop="0dp"
app:layout_behavior="@string/appbar_scrolling_view
_behavior">

<TextView
android:layout_width="match_
parent"
android:layout_height="match_
parent"
android:text="Update your
Profile"
android:textColor="@android:col
or/black"
android:textStyle="bold"
android:textSize="30sp"
android:layout_marginBottom="7
0dp" android:gravity="center"/>


<EditText
android:id="@+id/old_email"
android:layout_width="match_
parent"
android:layout_height="wrap_c
ontent"
android:hint="ConfirmEmail"
```

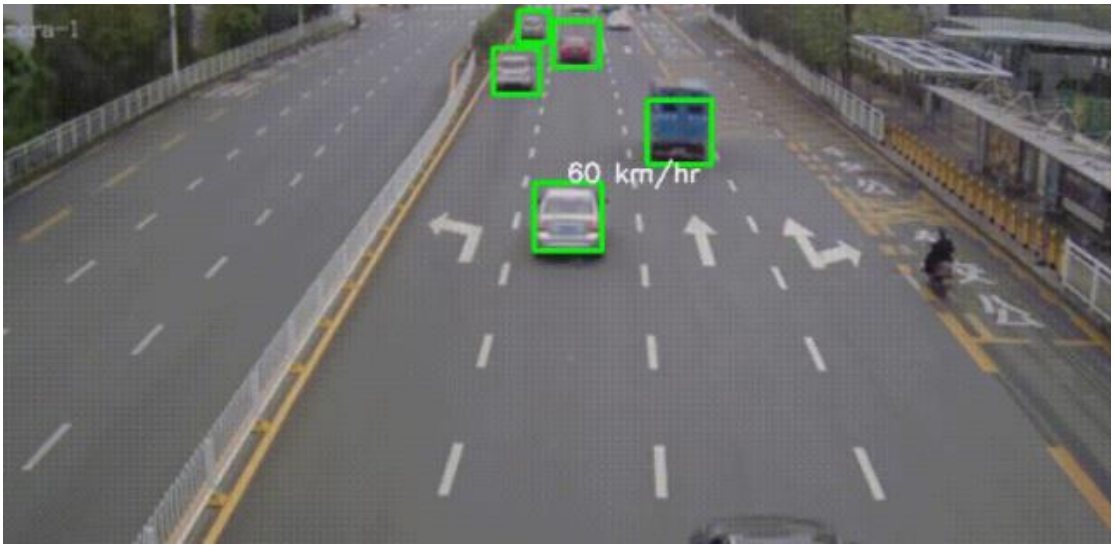
```
android:inputType="textEmailAd
dress" android:maxLines="1"
android:singleLine="true"/>

<EditText
android:id="@+id/new_email"
android:layout_width="match_
parent"
android:layout_height="wrap_c
ontent" android:hint="New
Email"
android:inputType="textEmailA
ddress" android:maxLines="1"
android:singleLine="true"/>

<EditText
android:id="@+id/password"
android:layout_width="match_
parent"
android:layout_height="wrap_c
ontent"
android:focusableInTouchMode
="true"
android:hint="@string/hint_pas
sword"
android:imeActionId="@+id/lo
gin"
android:imeOptions="actionUnsp
ecified"
android:inputType="textPasswor
d" android:maxLines="1"
android:singleLine="true"
```

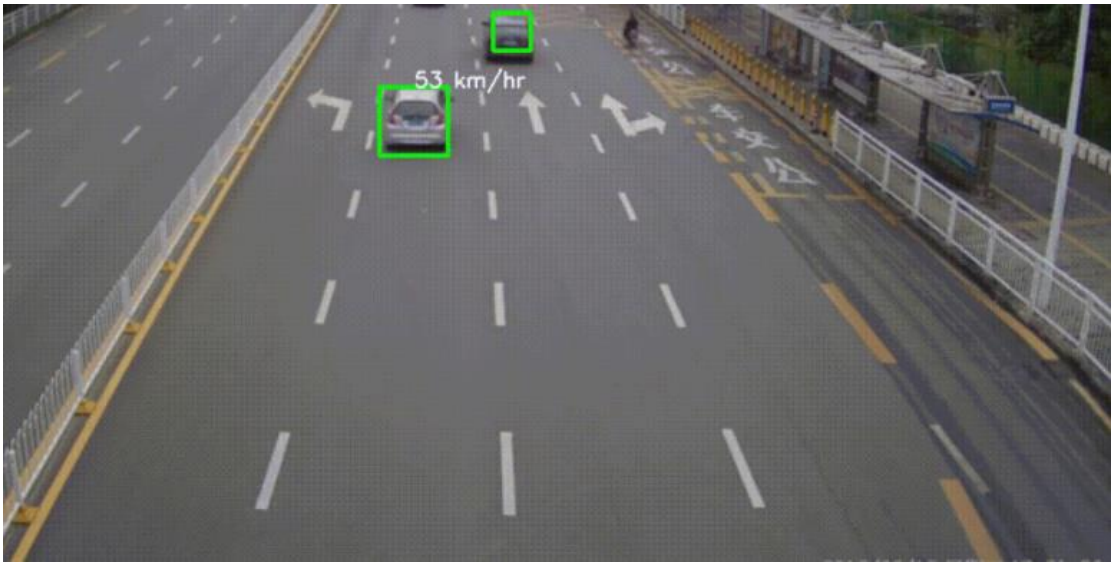
```
tools:ignore="InvalidImeActionId
"/>

<EditText
android:id="@+id/newPassword"
android:layout_width="match_pa
rent"
android:layout_height="wrap_con
tent"
android:focusableInTouchMode=
"true" android:hint="New
Password"
android:imeActionId="@+id/login
"
android:imeOptions="actionUnsp
ecified"
android:inputType="textPasswor
d" android:maxLines="1"
android:singleLine="true"
tools:ignore="InvalidImeActionId
"/>

<Button
android:id="@+id/s
end"
style="?android:textAppearanceSma
ll"
android:layout_width="wrap_conten
t"
android:layout_height="wrap_conte
nt"
android:layout_marginTop="16dp"
android:background="@android:col
```

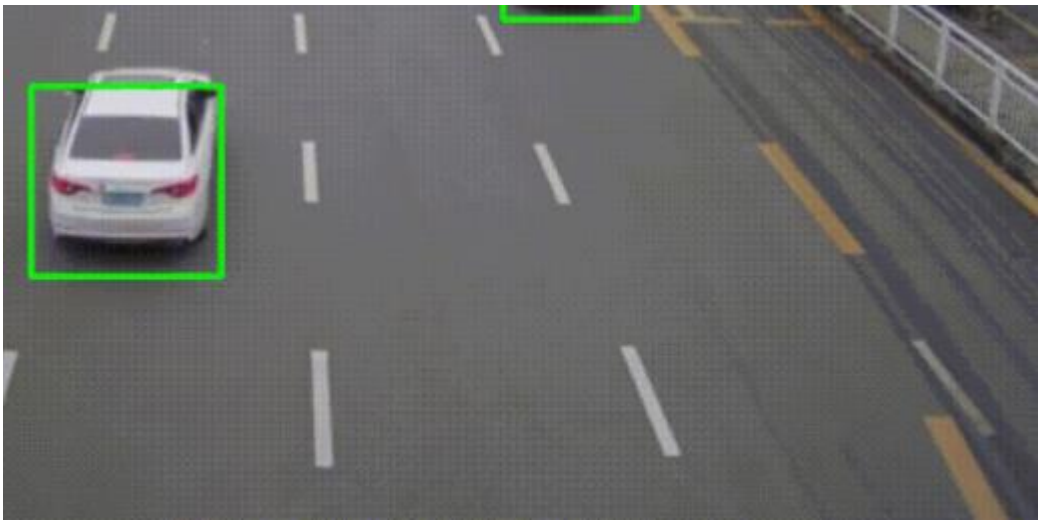
```
or/black" android:text="Send"
android:textColor="@android:color/
white" android:textStyle="bold"
android:layout_gravity="end"/>

<ProgressBar
android:id="@+id/progre
ssBar"
android:layout_width="3
0dp"
android:layout_height="3
0dp"
android:visibility="gone"
/>
<Button
android:id="@+id/re
move"
style="?android:textAppearanceSmall"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="16dp"
android:background="@color/colorPri
maryDark" android:text="Remove"
android:textColor="@android:color/whi
te" android:textStyle="bold" />
<android.support.design.widget.TextInp
utLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
>
<EditText
android:id="@+id/n
ame"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Name"
android:inputType="textCapWords"
android:maxLines="1"/>

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInp
utLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
>

<EditText
android:id="@+id/e
mail"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:hint="Email"
android:inputType="textEmailAddress"
android:maxLines="1"/>

</android.support.design.widget.TextInputLayout>
```

10. OUTPUT SCREENS (FORMS & REPORTS)







11. FUTURE ENHANCEMENT

In this paper, we introduced a model for tracking vehicles in traffic videos based on a detect-then-track paradigm, coupled with an optical-flow-based data-driven speed estimation approach, and described our solutions for Track 1 of the 2018 NVIDIA AI City Challenge. Our model performed well but was not as competitive as some of the other Challenge teams, displaying excessive variability. Due to lack of time, we did not compare our method against other detectthen-track algorithms, which we leave as future work. Additionally, we plan to investigate smoothing techniques for the predicted vehicle speeds which may lead to improved model performance.

12. CONCLUSION

In this paper, we introduced a model for tracking vehicles in traffic videos based on a detect-then-track paradigm, coupled with an optical-flow-based data-driven speed estimation approach, and described our solutions for Track 1 of the 2018 NVIDIA AI City Challenge. Our model performed well but was not as competitive as some of the other Challenge teams, displaying excessive variability. Due to lack of time, we did not compare our method against other detect then-track algorithms, which we leave as future work. Additionally, we plan to investigate smoothing techniques for the predicted vehicle speeds which may lead to improved model performance.

13. REFERENCES

[1] Data and Evaluation nvidia ai city challenge. https: //www.aicitychallenge.org/?page_id=9. Accessed: 2018-04-01.

[2] N. Bhandary, C. MacKay, A. Richards, J. Tong, and D. C. Anastasiu. Robust classification of city roadway objects for traffic related applications. 2017.

[3] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[4] B. Caprile and V. Torre. Using vanishing points for camera calibration. Int. J. Comput. Vision, 4(2):127–140, May 1990.

[5] A. Dutta, A. Gupta, and A. Zissermann. VGG image anno