NAME: RAJIV GUPTA

REG. NO: 18BCE2203

COURSE: SOFTWARE ENGINEERING LAB (L21+L22)

LAB ASSIGNMENT 2

PROJECT TITLE:

CHRONIC KIDNEY DISEASE PREDICTION MODEL

WORKBREAK DOWN STRUCTURE (WBS)

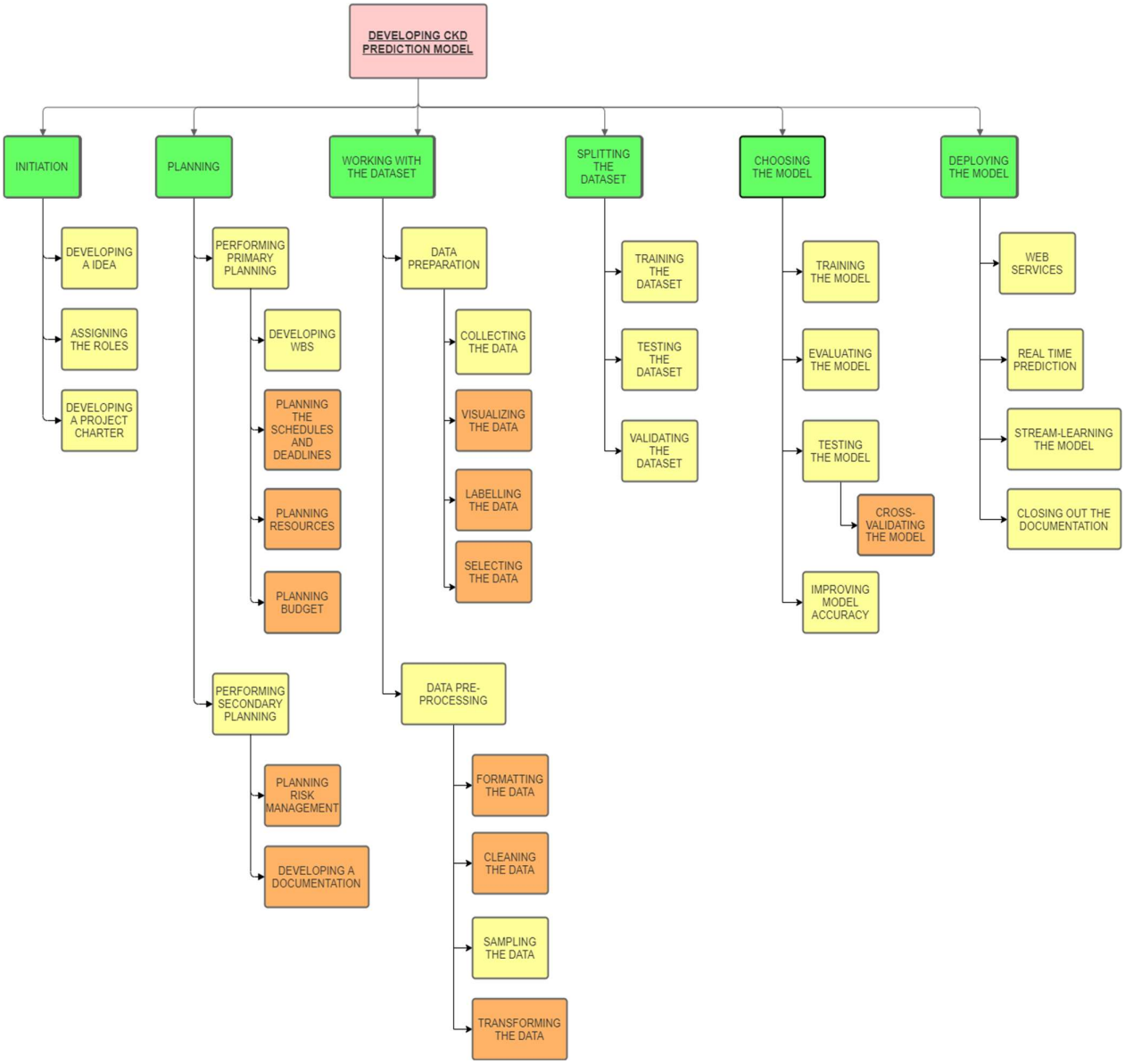
Breaking work into smaller tasks is a common productivity technique used to make the work more manageable and approachable. For projects, the **Work Breakdown Structure (WBS)** is the tool that utilizes this technique and is one of the most important project management documents. It singlehandedly integrates scope, cost and schedule baselines ensuring that project plans are in alignment.

- DIAGRAMS ARE DRAWN USING LUCIDCHART AND APP.DIAGRAMS.NET/DRAW.IO

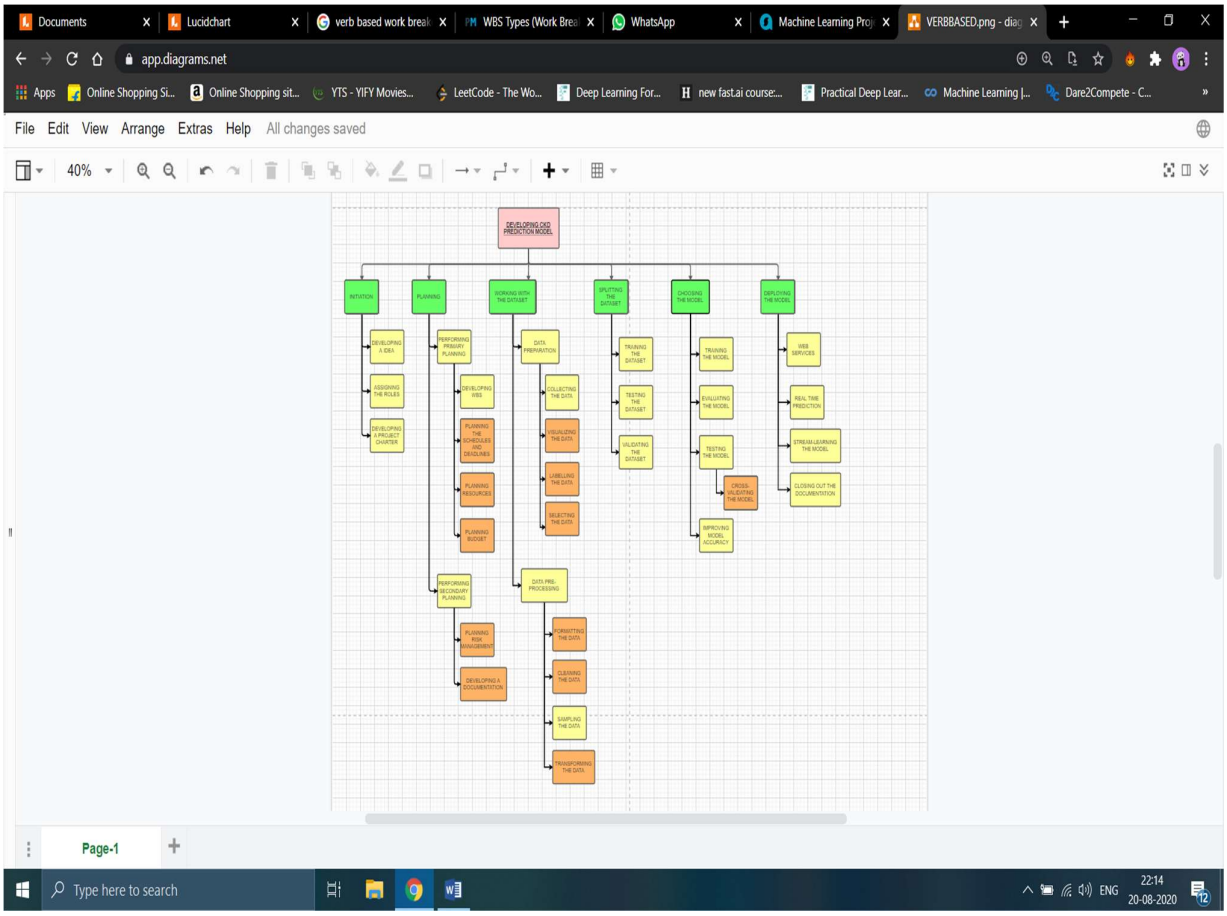
VERB BASED WORK-BREAKDOWN STRUCTURE

Verb-oriented WBS: a task-oriented WBS defines the deliverable of project work in terms of the actions that must be done to produce the deliverable. The first word in a given WBS element usually is a verb, such as, design, develop, optimize, transfer, test, etc.

FORMAT 1



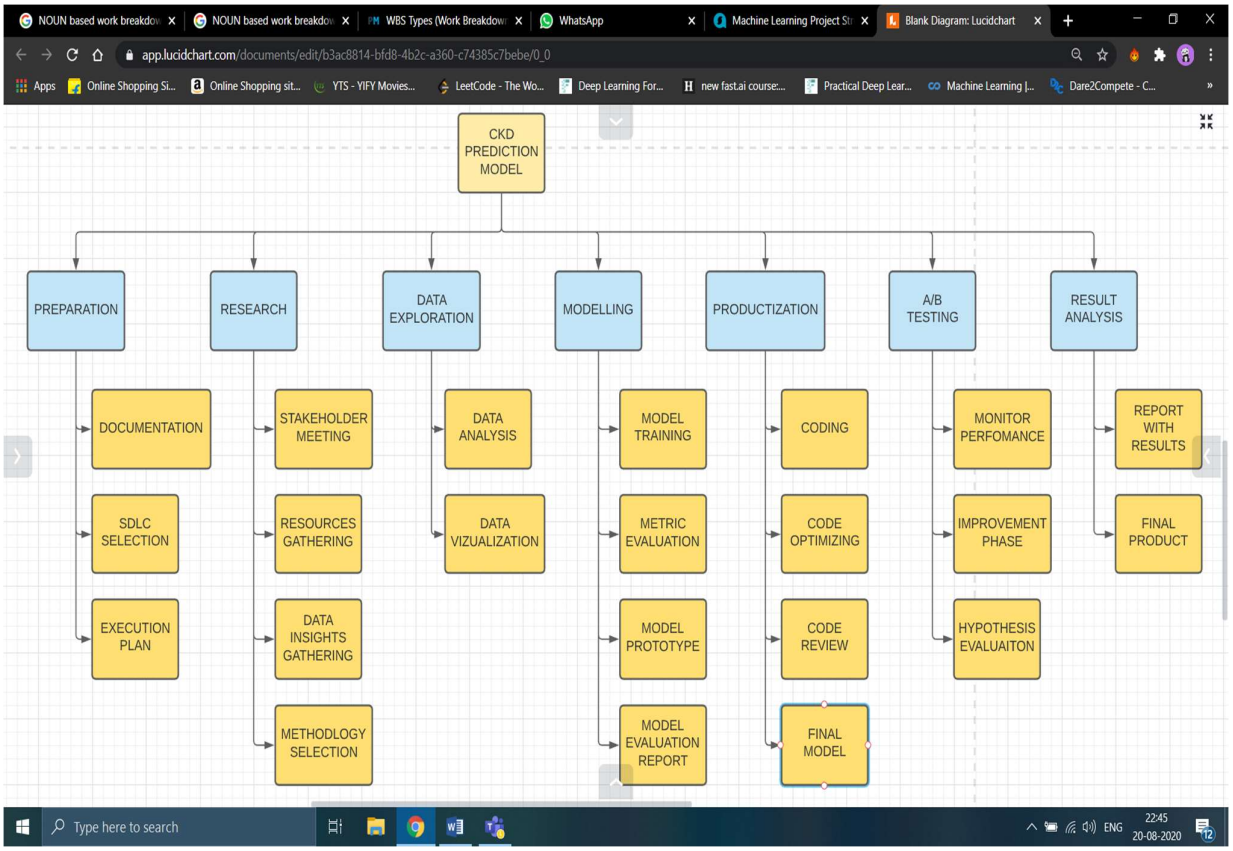
FORMAT 2


Documents
Lucidchart
verb based work break
WBS Types (Work Brea
WhatsApp
Machine Learning Proj
VERBBASED.png - diag
app.diagrams.net
Apps
Online Shopping Si...
Online Shopping sit...
YTS - YIFY Movies...
LeetCode - The Wo...
Deep Learning For...
new fast.ai course:...
Practical Deep Lear...
Machine Learning |...
Dare2Compete - C...
File Edit View Arrange Extras Help All changes saved
40%
DEVELOPING CKD PREDICTION MODEL
INITIATION
PLANNING
WORKING WITH THE DATASET
SPLITTING THE DATASET
CHOOSING THE MODEL
DEPLOYING THE MODEL
DEVELOPING A IDEA
ASSIGNING THE ROLES
DEVELOPING A PROJECT CHARTER
PERFORMING PRIMARY PLANNING
DEVELOPING WBS
PLANNING THE SCHEDULES AND DEADLINES
PLANNING RESOURCES
PLANNING BUDGET
PERFORMING SECONDARY PLANNING
PLANNING RISK MANAGEMENT
DEVELOPING A DOCUMENTATION
DATA PREPARATION
COLLECTING THE DATA
EQUALIZING THE DATA
LABELLING THE DATA
SELECTING THE DATA
DATA PRE-PROCESSING
FORMATTING THE DATA
CLEANING THE DATA
SAMPLING THE DATA
TRANSFORMING THE DATA
TRAINING THE DATASET
TESTING THE DATASET
VALIDATING THE DATASET
TRAINING THE MODEL
EVALUATING THE MODEL
TESTING THE MODEL
CROSS VALIDATING THE MODEL
IMPROVING MODEL ACCURACY
WEB SERVICES
REAL TIME PREDICTION
STREAM LEARNING THE MODEL
CLOSING OUT THE DOCUMENTATION
Page-1
Type here to search
ENG
22:14
20-08-2020

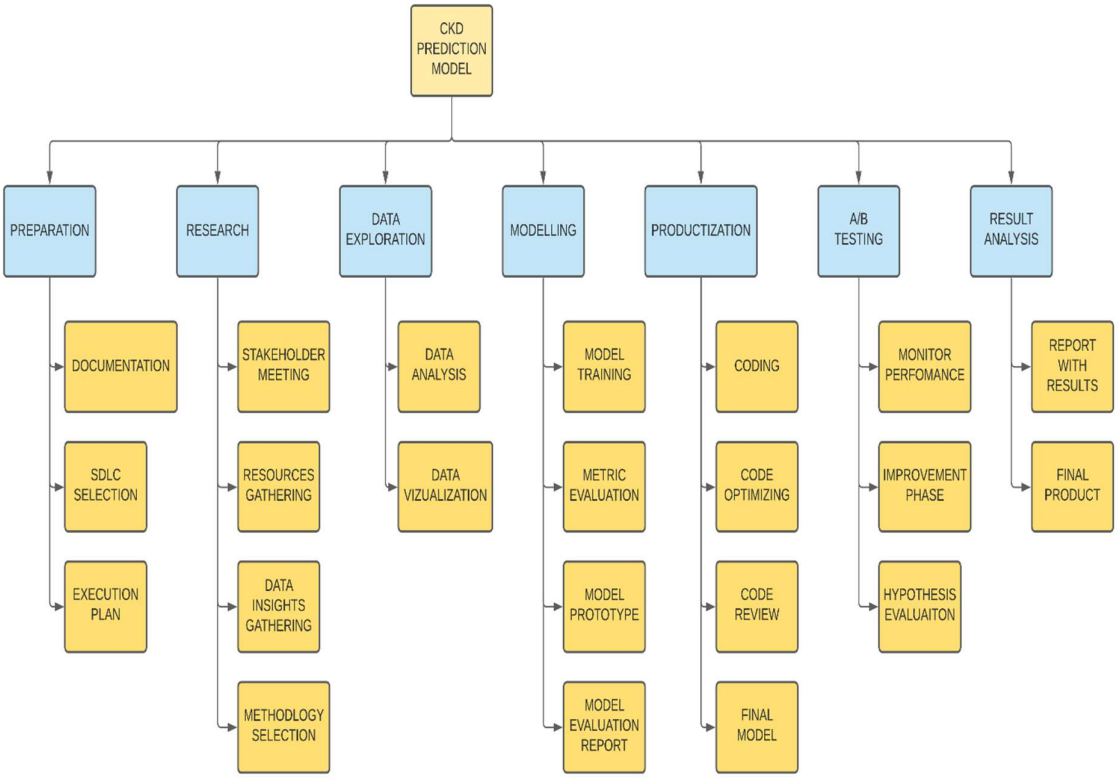
NOUN BASED WORK-BREAKDOWN STRUCTURE

Noun-oriented WBS: a deliverable-oriented WBS defines project work in terms of the components (physical or functional) that make up the deliverable. In this case, the first word in a given WBS element is a noun, such as, Module A, Subsystem A, Automobile Engine, Antenna, etc. Since the nouns are usually parts of a product, this WBS type is sometimes called a “Product Breakdown Structure (PBS). Deliverable-oriented WBS structures are the preferred type according to PMI’s definition.

FORMAT 1



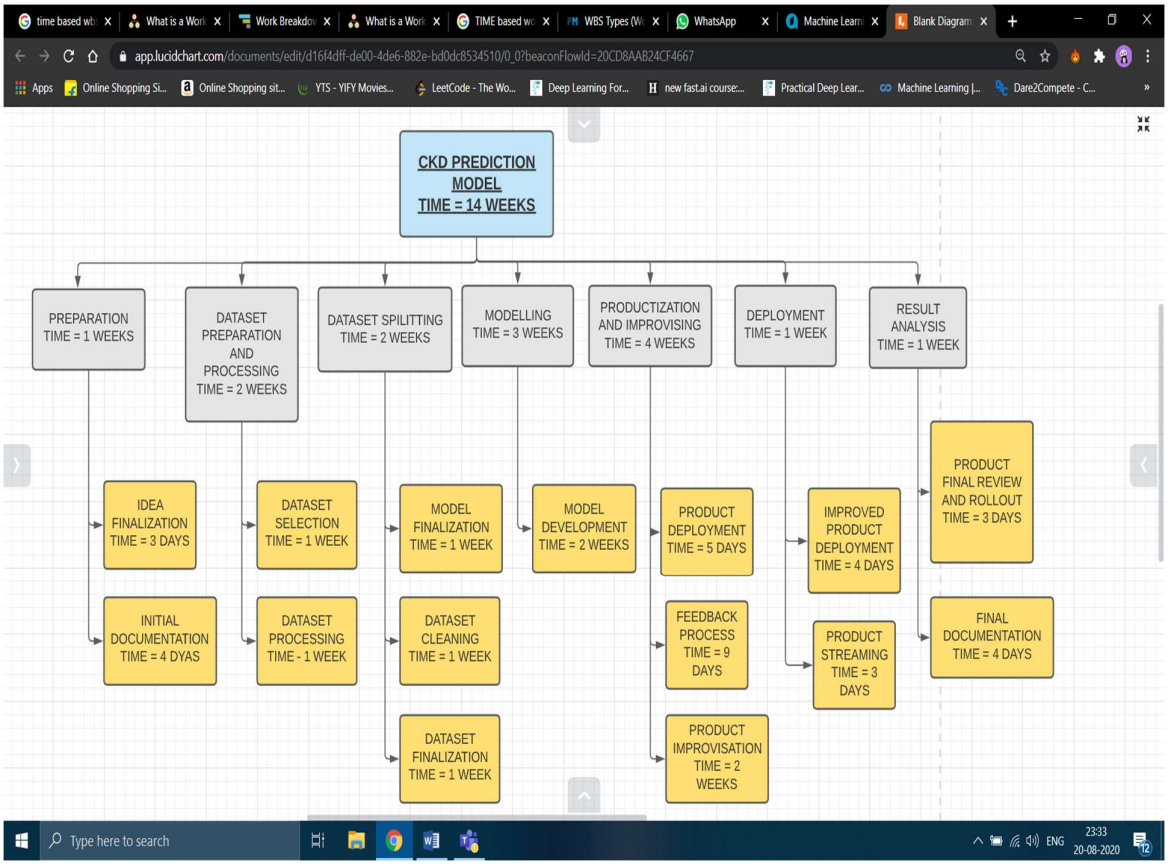
FORMAT 2


CKD PREDICTION MODEL
PREPARATION
RESEARCH
DATA EXPLORATION
MODELLING
PRODUCTIZATION
A/B TESTING
RESULT ANALYSIS
DOCUMENTATION
SDLC SELECTION
EXECUTION PLAN
STAKEHOLDER MEETING
RESOURCES GATHERING
DATA INSIGHTS GATHERING
METHODLOGY SELECTION
DATA ANALYSIS
DATA VIZUALIZATION
MODEL TRAINING
METRIC EVALUATION
MODEL PROTOTYPE
MODEL EVALUATION REPORT
CODING
CODE OPTIMIZING
CODE REVIEW
FINAL MODEL
MONITOR PERFOMANCE
IMPROVEMENT PHASE
HYPOTHESIS EVALUAITON
REPORT WITH RESULTS
FINAL PRODUCT

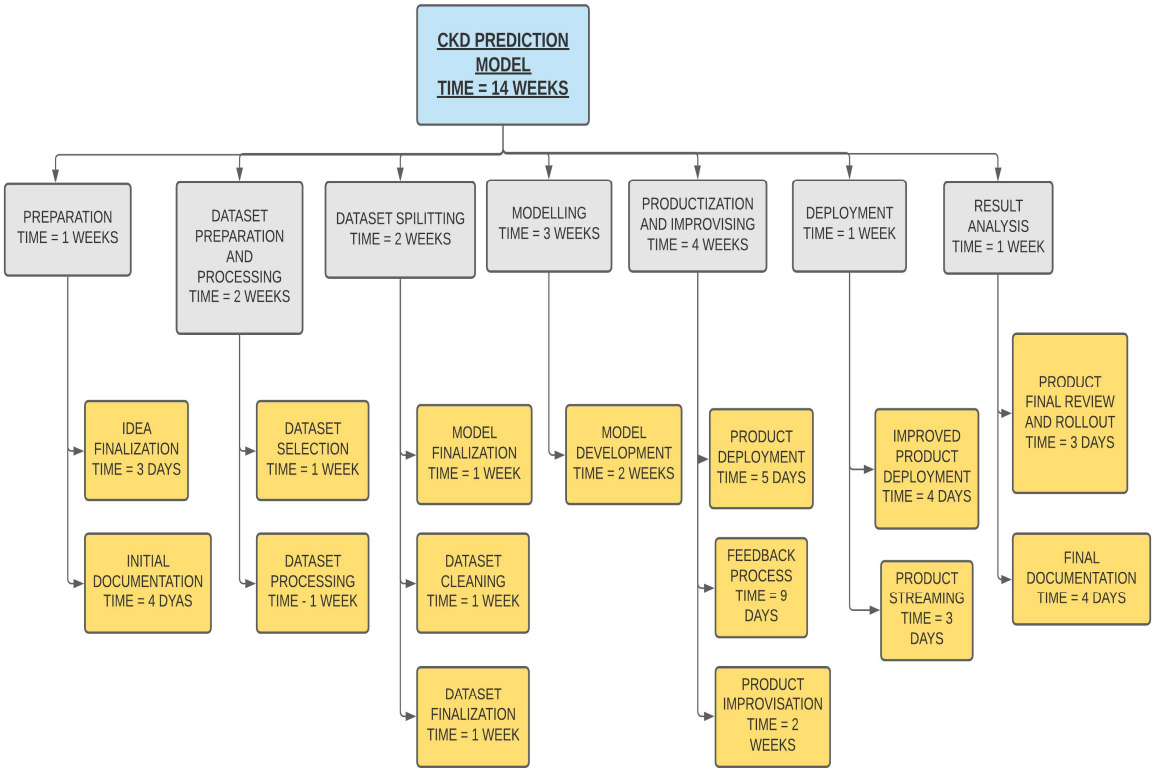
TIME-PHASED WORK-BREAKDOWN STRUCTURE

Time-phased WBS: a “time-phased” WBS is one that is used on very long projects. It breaks the project into major phases instead of tasks. In this type, a “rolling wave” approach is adopted and only the near-term phase is planned in detail.

FORMAT 1



FORMAT 2


CKD PREDICTION MODEL
TIME = 14 WEEKS
PREPARATION
TIME = 1 WEEKS
DATASET PREPARATION AND PROCESSING
TIME = 2 WEEKS
DATASET SPILITTING
TIME = 2 WEEKS
MODELLING
TIME = 3 WEEKS
PRODUCTIZATION AND IMPROVISING
TIME = 4 WEEKS
DEPLOYMENT
TIME = 1 WEEK
RESULT ANALYSIS
TIME = 1 WEEK
IDEA FINALIZATION
TIME = 3 DAYS
INITIAL DOCUMENTATION
TIME = 4 DYAS
DATASET SELECTION
TIME = 1 WEEK
DATASET PROCESSING
TIME - 1 WEEK
MODEL FINALIZATION
TIME = 1 WEEK
DATASET CLEANING
TIME = 1 WEEK
DATASET FINALIZATION
TIME = 1 WEEK
MODEL DEVELOPMENT
TIME = 2 WEEKS
PRODUCT DEPLOYMENT
TIME = 5 DAYS
FEEDBACK PROCESS
TIME = 9 DAYS
PRODUCT IMPROVISATION
TIME = 2 WEEKS
IMPROVED PRODUCT DEPLOYMENT
TIME = 4 DAYS
PRODUCT STREAMING
TIME = 3 DAYS
PRODUCT FINAL REVIEW AND ROLLOUT
TIME = 3 DAYS
FINAL DOCUMENTATION
TIME = 4 DAYS

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

We will be using **RAPID APPLICATION DEVELOPEMNT MODEL (RAD)**

The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

Rapid application development is a software development methodology that uses minimal planning in favour of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.

RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

RAD Model Design

RAD model distributes the analysis, design, build and test phases into a series of short, iterative development cycles.

Following are the various phases of the RAD Model −

Business Modelling

The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

Data Modelling

The information gathered in the Business Modelling phase is reviewed and analysed to form sets of data objects vital for the business. The attributes of all data sets is identified and defined. The relation between these data objects are established and defined in detail in relevance to the business model.

Process Modelling

The data object sets defined in the Data Modelling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined in this phase. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

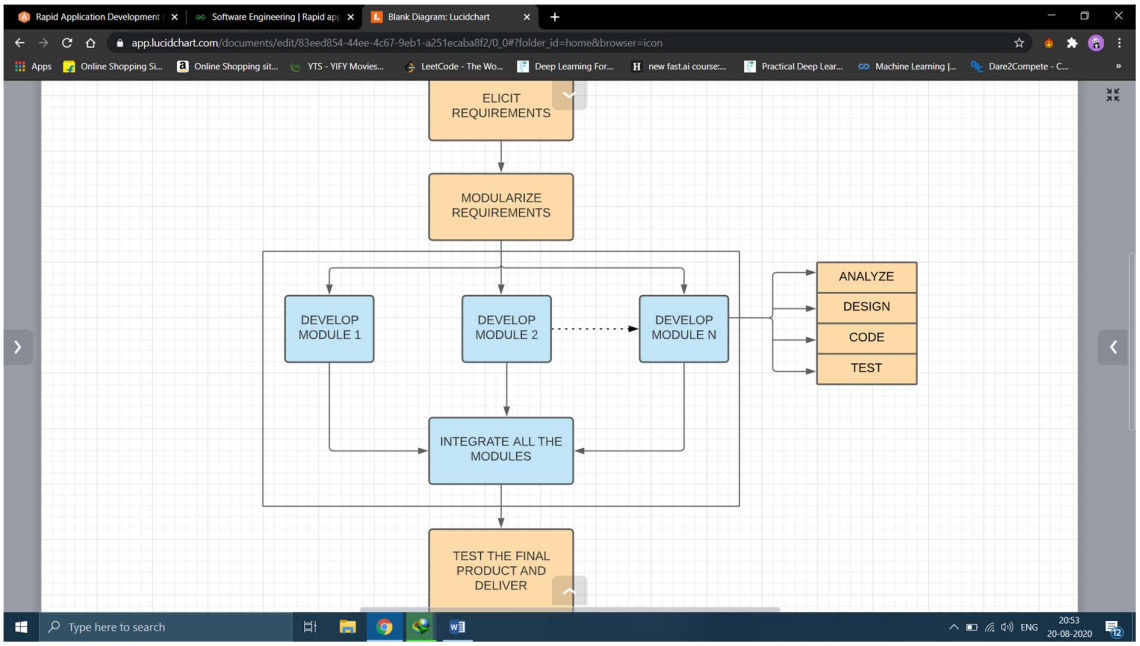
Application Generation

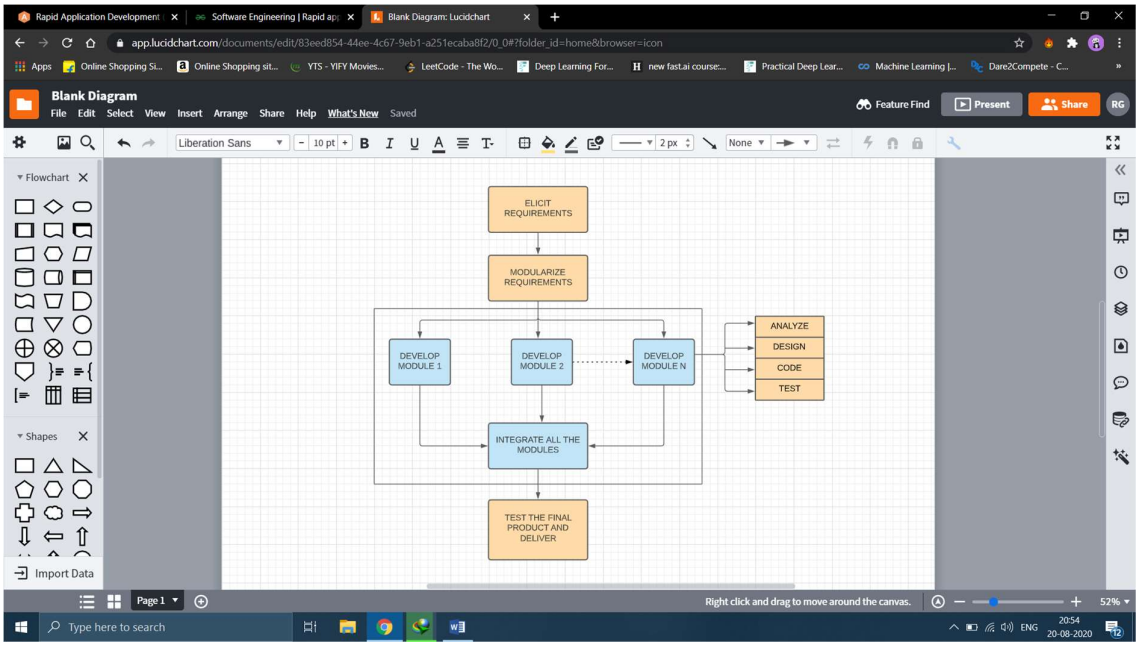
The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

Testing and Turnover

The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues.

RAD MODEL DIAGRAM





Advantages of RAD Model

- Measurable Progress: With frequent iterations, components, and prototypes coming down the pipe, progress on the overall project, as well as lesser segments, can be easily measured and evaluated to maintain schedules and budgets.
- Quickly Generate Productive Code: As a larger percentage of active software developers move into multi-discipline roles (i.e. full-stack developers), a RAD methodology allows skilled team members to quickly produce prototypes and working code to illustrate examples that might otherwise take weeks or months to see the light of day using a slower development technique.
- Compartmentalization of System Components: Much in the same way that object-oriented programming practices keep objects and components quarantined from one another, RAD inherently has the same beneficial impact on the components generated during development. By forcing designers and developers to create components that are functional and independent on their own, to be used in an iterative release or prototype, each element within the overall software system is compartmentalized and therefore easily modified as the needs of the software evolve.
- Rapid, Constant User Feedback: As discussed above, obtaining relevant user feedback during development is invaluable. RAD methodologies allow for near-constant user interfacing and feedback through frequent iterations and prototype releases, giving the entire team priceless evaluation and criticism when it's needed most.
- Early Systems Integration: While most waterfall method software projects must, by their very nature, wait until the tail end of the lifecycle to begin integrations with other systems or services, a rapidly developed application becomes integrated almost immediately. By requiring early integrations within a prototype, a RAD system quickly identifies any errors or complications within integrations and forces immediate resolutions.
- Simple Adaptability: During development, software is a fairly malleable form. Since code can be changed that dramatically alters the entire system or generates new components, it is to the advantage of the development team to make use of this flexibility early and often, by iterating and prototyping potential concepts or ideas throughout development.

Disadvantages of RAD Model

- Requires Modular Systems: Since each component within the system should be iterable and testable on its own accord, the overall system design when using RAD requires that each component be modular, allowing elements to be swapped in and out or altered by a variety of team members.
- Difficulty Within Large-Scale Projects: While rapid application development methods lead to far greater flexibility during the design and development process, it will also tend to reduce control and restrictions. While this isn't inherently negative, properly managing this added flexibility and volatility within the scope of the whole project can be difficult for larger applications.
- Demands Frequent User Interfacing: Gaining user insight and feedback early and often is certainly a benefit from a design perspective, but this double-edged sword requires that the team be both willing and able to communicate with users on a much more frequent basis, in comparison to a typical waterfall development method.
- Depends Upon Skilled Developers: While many developers these days are multi-disciplined, it is worth noting that use of RAD techniques does require a greater overall skill across the development team, in order to quickly adapt as the system and components evolve.

JUSTIFICATION FOR USING RAD MODEL

1. We use RAD model because there is a need to create a system that can be modularized in 2-3 months of time.
2. Also it increases the reusability of components and review of our work can be done quickly.
3. The other main purpose of using this model is because it allows integration from very beginning which solves a lot of integration issues.
4. Due to limited amount of time and man power, this will allow us to deliver our project in small pieces, that is whole project can be divided into number of smaller components.
5. The technical risks are low and also the requirements of the product are known well beforehand.
6. Also there is a good scope of getting reliable feedback on our deliverables for further improvement.