

# MINI PROJECT

TITLE: AIRPORT CHECK-IN SYSTEM USING HAND GESTURES

Date of Performance	Date of Submission	Marks (10)					Sign / Remark
		A	B	C	D	E	
		2	3	2	2	1	
		Total Marks					

# **AIRPORT CHECK-IN SYSTEM USING HAND GESTURES**

Submitted in partial fulfilment of requirement of

**University of Mumbai**

For the Degree of

**Third Year of Computer Engineering (Sem VI)**

Submitted By

**Sakshi Kodarlikar (32)**

**Shruti Marathe (38)**

**Soham Patil (55)**

Under Guidance of

**Prof. Deepti Vijay Chandran**



Department of Computer Engineering

**Smt. Indira Gandhi College of Engineering**

Affiliated to University of Mumbai

(2020-2021)

## **Project Report Approval Certificate**

**This Project report entitled Airport Check-in using Hand Gestures**

By

Sakshi Kodarlikar (32)

Shruti Marathe (38)

Soham Patil (55)

Is approved for Mini Project lab, Sem VI Computer Engineering.

Date:

Place: Ghansoli

-----  
External Examiner

-----  
Prof. Deepti V. Chandran  
Internal Guide

-----  
Prof. Sonali Deshpande  
Head of Department

-----  
Dr. Sunil Chavan  
Principal

## Declaration

We declare that this written submission represents our own ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any act/data/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.



-----  
(Sakshi Kodarlikar(32))



-----  
(Shruti Marathe(38))



-----  
(Soham Patil(55))

## ABSTRACT

**Title:** Airport check-in using hand gestures

Touch less interfaces are even more critical in the era of COVID-19. We're wary of touch like never before, an aversion likely to persist until a viable vaccine becomes available – and potentially beyond. Reducing or eliminating touch points is essential for safely reopening businesses. The pandemic has crystallized the value proposition of touch less technologies, accelerating their development and deployment. A desire for contactless sensing and hygiene concerns are the top drivers of demand for touch less technology. As the pandemic has swept over the world, gesture control and touch less user interfaces have become a hot topic. Both provide the ability to interact with devices without physically touching them. The global COVID-19 pandemic has created a greater need for alternatives to common daily practices to reduce the spread of the virus. During the coronavirus pandemic, it's not surprising that people are reluctant to use touchscreens in public places. In this scenario, it is good to use touch less technologies. Because touch-based technologies can cause the spread of virus. People are more tend to use touch less systems as much as possible in order to reduce the spread of virus. Airlines have switched to self-service check-in, which is touch-based system. It's not safe to use that services anymore. There's never been a need for touch less tech like there's a need for it now. With the help of Tensor Flow model, we will be detecting hand points and identifying the different gestures. To implement gesture-based control instead of hand check-in kiosk.

## **List of Abbreviations**

- 1) BE: Bachelor of Engineering
- 2) DFD: Data Flow Diagram
- 3) AI: Artificial Intelligence
- 4) NN: Neural Network
- 5) UI: User Interface

## List of Figures

1) FIG NO. 3.1 – Prototype Model	(Page 11)
2) FIG NO. 3.2 – Timeline chart	(Page 10)
3) FIG NO 4.1 – DFD level 0	(Page 13)
4) FIG NO 4.2 – DFD level 1	(page 13)
5) FIG NO. 5.1 – Flow Diagram	(Page 16)
6) FIG NO. 5.2 – Use Case Diagram	(Page 17)
7) FIG NO. 5.3 – Activity Diagram	(Page 18)
8) FIG NO. 5.4 – Sequence Diagram	(Page 19)
9) FIG NO. 5.5 – Class Diagram	(Page 20)
10) FIG NO 6.1 – Creating electron app	(page 22)
11) FIG NO 6.2 – Installing packages	(page 22)
12) FIG NO 6.3 – UI - how to use gestures	(page 23)
13) FIG NO 6.4 – UI - seat selection	(page 23)
14) FIG NO 6.5 – Loading model	(page 24)
15) FIG NO 6.6 – 21 3D landmarks plotted on hand by mediapipe	(page 24)
16) FIG NO 6.7 – Selecting and displaying the centroid of hand	(page 25)
17) FIG NO 6.8 – Plotting landmarks of hand	(page 25)
18) FIG NO 6.9 – Training the model for pinch gesture	(page 26)
19) FIG NO 6.10 – Setting the click events	(page 27)
20) FIG NO 7.1 – Hand detection with 21 3D landmark	(page 30)
21) FIG NO 7.2 – Typing using hand gestures	(page 30)

# INDEX

Chapter No.	Contents	Page No.
	<b>Abstract</b>	<b>i</b>
	<b>List of Abbreviations</b>	<b>ii</b>
	<b>List of Figures</b>	<b>iii</b>
<b>1</b>	<b>INTRODUCTON</b>	<b>1</b>
	1.1 Problem Statement	<b>2</b>
	1.2 Objective	<b>2</b>
	1.3 Scope	<b>2</b>
	1.4 Report Organization	<b>2</b>
<b>2</b>	<b>REVIEW OF LITERATURE</b>	<b>3</b>
	2.1 Research paper analysis	<b>4</b>
<b>3</b>	<b>PLANNING AND FORMULATION</b>	<b>7</b>
	3.1 Project Development Model	<b>8</b>
	3.2 Feasibility study	<b>10</b>
	3.3 Timeline chart	<b>10</b>
<b>4</b>	<b>REQUIREMENT ANALYSIS</b>	<b>11</b>
	4.1 Hardware Requirements	<b>12</b>
	4.2 Software Requirements	<b>12</b>
	4.3 Functional Requirements	<b>12</b>
	4.4 Non – Functional Requirements	<b>12</b>
	4.5 Data Flow Diagram	<b>13</b>
	4.6 Project Methodology	<b>13</b>
<b>5</b>	<b>PROPOSED SYSTEM</b>	<b>15</b>
	5.1 System Design	<b>16</b>
	5.1.1 Flow Diagram	<b>16</b>
	5.1.2 Use case diagram	<b>17</b>
	5.1.3 Activity diagram	<b>18</b>
	5.1.4 Sequence diagram	<b>19</b>
	5.1.5 Class diagram	<b>20</b>
	<b>IMPLEMENTATION</b>	<b>21</b>
<b>6</b>	6.1 Creating UI	<b>22</b>
	6.2 Selection of Model	<b>24</b>
	6.3 Implementing the Model	<b>25</b>



	6.4 Training Model	<b>26</b>
	6.5 Setting Click Events	<b>27</b>
	6.6 Algorithm	<b>28</b>
<b>7</b>	<b>RESULT</b>	<b>29</b>
	<b>CONCLUSION</b>	<b>31</b>
<b>8</b>	8.1 Conclusion	<b>32</b>
	8.2 Future scope	<b>32</b>
	<b>REFERENCES</b>	<b>33</b>
	<b>ACKNOWLEDGEMENTS</b>	<b>34</b>

## **CHAPTER 1**

### **INTRODUCTION**

# 1. INTRODUCTION

As we all know, world is currently suffering from the crisis of COVID – 19 virus. Touching any random object can cause major harm in anyone's life. To minimize the spread of any virus, nowadays people are more tend to use touch less systems as much as possible. We know that almost all the airlines have switched to self-service check-in, which basically based on touch-based system. In current scenario, it's not safe to use that service as there are lots of people who come in contact with the machine. In the last few years, gesture-controlled interactive surfaces have become widespread. So, our project aims to make the airport check-in machine usable with the help of gestures i.e., try to build touch less model with the help of hand gestures.

## 1.1 PROBLEM STATEMENT

Developing airport check-in software which is capable of working using hand recognition and gestures to control it rather than using touch based controls.

## 1.2 OBJECTIVE

- 1) To make the existing touch based check-in kiosk work with gestures to make it touch less.
- 2) Confirmation of certain rules and regulations using hand gestures.
- 3) Booking ID and seat number using hand gestures.

## 1.3 SCOPE

With the use of TensorFlow hand pose model, touch less check-in kiosk has been made which works with hand gestures. We have used 2 hand gestures in this project. Pinching with thumb and finger will click/select desired option. Centroid of palm taken from the 21 3D landmarks of hand is used as cursor.

## 1.4 REPORT ORGANIZATION

- **In Chapter 2**, we will see the Planning and formulation of project is given, usage of Spiral model and how we integrated and worked around the model.
- **In Chapter 3**, we will see what are the hardware and software requirements for the project as well as what are the tools needed to be installed.
- **In Chapter 4**, we will see the system proposed is introduced which will tell the deep specification which of the project and will tell how the different modules of system will work, the flow of the project regarding data flow, control flow and other flow of the system.
- **In Chapter 5**, we will see the implementation of the project and the screenshots of the model working.
- **In Chapter 6**, we will display the results accordingly.
- **In Chapter 7**, conclusion and the future scope is mentioned.

## **CHAPTER 2**

### **REVIEW OF LITERATURE**

## 2. REVIEW OF LITERATURE

### 2.1 Research paper analysis

- ❑ Dharmik Bhanushali, Dipanshu Someshwar, Vismay Chaudhari, Ms. Bhargavi Dalal (2020). Hand Gestures Recognition using Deep Learning. <https://www.irjet.net/archives/V7/i3/IRJET-V7I31032>

The use of Deep Learning over Machine Learning is done because Machine Learning shows output accuracy in form of 0 or 1, whereas Deep Learning provides better numerical accuracy results that ranges from 0 to 1 [6]. Since the proposed project demands high need of accuracy in Classification Algorithms it is better to make use of Deep Learning. Just like Data Classification is the most important aspect of Data Science, similarly Image Classification plays a very big role in Computer Vision. Image Classification first deals with image preprocessing, then it executes image segmentation and then it deals with key feature extraction that helps for finding uniqueness in different trained images and lastly executes matching identification

- ❑ B., Abhishek & Krishi, Kanya & M., Meghana & Daaniyaal, Mohammed & S., Anupama. (2020). Hand gesture recognition using machine learning algorithms. Computer Science and Information Technologies. 1. 116-120. 10.11591/csit.v1i3.p116-120.

The implementation is divided into four main steps: 1. Image Enhancement and Segmentation 2. Orientation Detection 3. Feature Extraction 4 [1]. Classification. This work was focused on above four categories but main limitation was change of color was happening very rapidly by the change in the different lighting condition, which may cause error or even failures. For example, due to insufficient light condition, the existence of hand area is not detected but the non-skin regions are mistaken for the hand area because of same color [2]. Involves three main steps for hand gesture recognition system: 1. Segmentation 2. Feature Representation 3. Recognition Techniques. The system is based on Hand gesture recognition by modeling of the hand in spatial domain. The system uses various 2D and 3D geometric and non-geometric models for modeling.

- ❑ Fan Zhang Valentin Bazarevsky Andrey Vakunov Andrei Tkachenka George Sung Chuo-Ling Chang Matthias Grundmann.( 2020). MediaPipe Hands: On-device Real-time Hand Tracking Hand tracking is a vital component to provide a natural way for interaction and communication in AR/VR, and has been an active research topic in the industry [2] [15]. Vision-based hand pose estimation has been studied for many years. A large portion of previous work requires specialized hardware, e.g. depth sensors [13][16][17][3][4]. Other solutions are not lightweight enough to run real-time on commodity mobile devices[5] and thus are limited to platforms equipped with powerful processors. In this pa per, we propose a novel solution that does not require any additional hardware and performs in real-time on mobile devices. Our main contributions are: • An efficient two-stage hand tracking pipeline that can track multiple hands in real-time on mobile devices. • A hand pose estimation model that is capable of predicting 2.5D hand pose with only RGB input. • And open source hand tracking pipeline as a ready-to go solution on a variety of platforms, including An droid, iOS, Web (Tensorflow.js[7]) and desktop PCs.

With MediaPipe[12], our hand tracking pipeline can be built as a directed graph of modular components, called Calculators. Mediapipe comes with an extensible set of Calculators to solve tasks like model inference, media processing, and data transformations across a wide variety of devices and platforms. Individual Calculators like cropping, rendering and neural network computations are further optimized to utilize GPU acceleration. For example, we employ TFLite GPU inference on most modern phones. Our MediaPipe graph for hand tracking is shown in Figure 5. The graph consists of two subgraphs one for hand detection and another for landmarks computation. One key optimization MediaPipe provides is that the palm detector only runs as needed (fairly infrequently), saving significant computation. We achieve this by deriving the hand location in the current video frames from the computed hand landmarks in the previous frame, eliminating the need to apply the palm detector on every frame. For robustness, the hand tracker model also outputs an additional scalar capturing the confidence that a hand is present and reasonably aligned in the input crop. Only when the confidence falls below a certain threshold is the hand detection model reapplied to the next frame.

Our hand tracking solution can readily be used in many applications such as gesture recognition and AR effects. On top of the predicted hand skeleton, we employ a simple algorithm to compute gestures, see Figure 6. First, the state of each finger, e.g. bent or straight, is determined via the accumulated angles of joints. Then, we map the set of finger states to a set of predefined gestures. This straightforward, yet effective technique allows us to estimate basic static gestures with reasonable quality. Beyond static gesture recognition, it is also possible to use a sequence of landmarks to predict dynamic gestures. Another application is to apply AR effects on top of the skeleton. Hand based AR effects currently enjoy high popularity. In Figure 7, we show an example AR rendering of the hand skeleton in neon light style.

Our hand tracking solution utilizes an ML pipeline consisting of two models working together:

- A palm detector that operates on a full input image and locates palms via an oriented hand bounding box.
- A hand landmark model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks.

Providing the accurately cropped palm image to the hand landmark model drastically reduces the need for data augmentation (e.g. rotations, translation and scale) and allows the network to dedicate most of its capacity towards landmark localization accuracy. In a real-time tracking scenario, we derive a bounding box from the landmark prediction of the previous frame as input for the current frame, thus avoiding applying the detector on every frame. Instead, the detector is only applied on the first frame or when the hand prediction indicates that the hand is lost.

- ❑ Sun, J.-H., Ji, T.-T., Zhang, S.-B., Yang, J.-K., & Ji, G.-R. (2018). Research on the Hand Gesture Recognition Based on Deep Learning. 2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE). doi:10.1109/isape.2018.8634348

The hand gesture segmentation based on skin color model is susceptible to being interrupted by the objects with the similar color of hands, such as human face and so on. In order to overcome the above shortcomings, the hand gesture segmentation based on model features is adopted after detection of skin color. Then, the hand gestures features are extracted by a great deal of sample of

hand gestures and the classifiers are trained by using these features, which is conducive to distinguishing the hand area and non-hand area. The paper adopts the AdaBoost classifier based on Haar feature. Haar feature reflects the image grayscale value change. The black and white rectangle areas are used to compose the feature model. In feature model, the pixel sums under white areas are subtracted from the pixel sums under the black areas, and express the feature value of objects. As shown in Fig.3, A and B represent margin feature, C presents linear feature while D represents diagonal feature. Lienhart R. et al.[7] expand the above basic features and form the expanded rectangle feature. A B C D Fig.3 Basic haar feature If all the rectangle feature areas are traversed when calculating Haar feature value, it will cause much repeated calculation and waste a lot of time. Two integral images can rapidly calculate the rectangle feature and its major idea is to convert the original image into integral image. When calculating the sum of pixel within one specific area, just the values of four angular points of the matrix area in the integral image are indexed, then the simple addition and minus calculation is required for obtain the Haar feature value.

## **CHAPTER 3**

### **PLANNING AND FORMULATION**



### 3. PLANNING AND FORMULATION

#### 3.1 Project Development Model

##### PROTOTYPE MODEL

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possibly exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

##### Steps of Prototype Model

1. Requirement Gathering and Analyst
2. Quick Decision
3. Build a Prototype
4. Assessment or User Evaluation
5. Prototype Refinement
6. Engineer Product

##### Advantage of Prototype Model

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

##### Disadvantage of Prototype Model

1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
  - Costs customer money
  - Needs committed customer
  - Difficult to finish if customer withdraw
  - May be too customer specific, no broad market

3. Difficult to know how long the project will last.
4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.

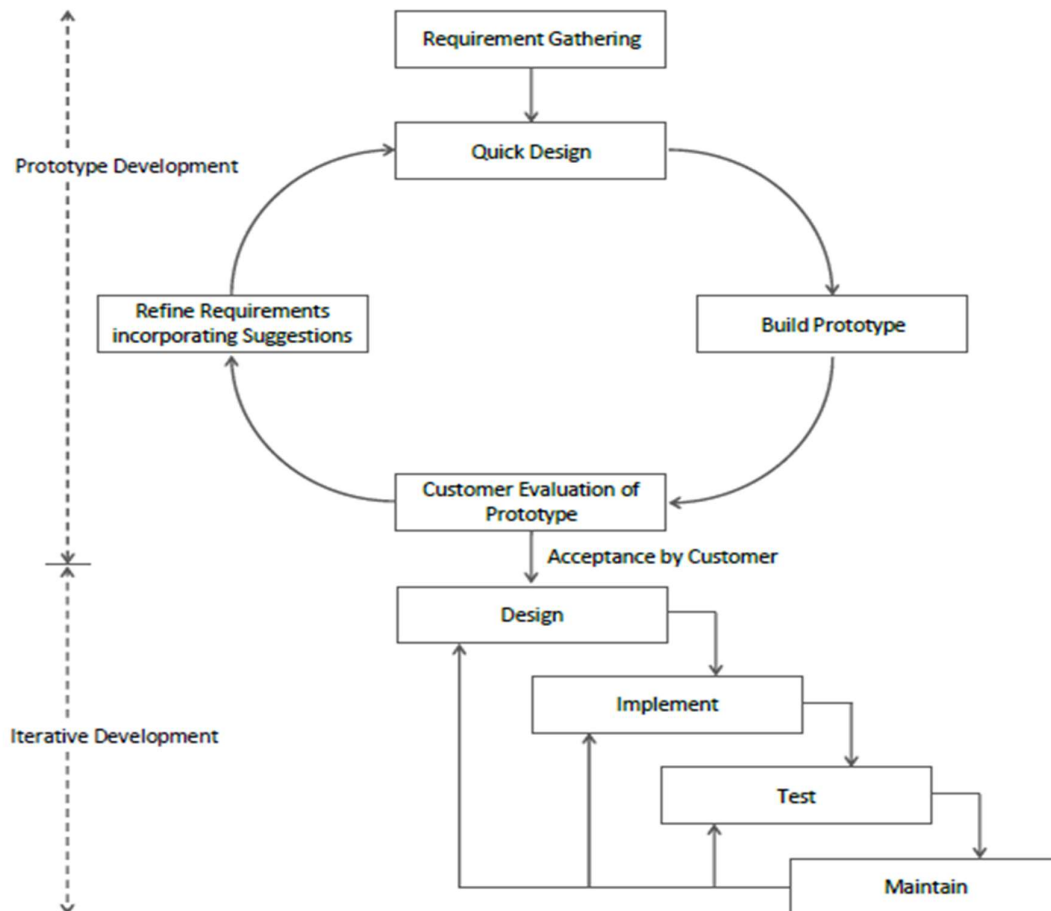


FIG NO 3.1 – Prototype model

### 3.2 Feasibility Study

A feasibility study is an analysis that takes all of a project's relevant factors into account—including economic, technical, legal, and scheduling considerations—to ascertain the likelihood of completing the project successfully. Project managers use feasibility studies to discern the pros and cons of undertaking a project before they invest a lot of time and money into it. Feasibility studies also can provide a company's management with crucial information that could prevent the company from entering carelessly into risky businesses.

#### Technical Feasibility:

The project is built on electron so it is highly maintainable. It is based on a single core unit, so if one unit stops processing the other still functions properly and database can be managed easily. The backend is highly modifiable to store data in different types of database. The application can work efficiently in remote and private network connection.

#### Economic Feasibility:

The project economic implementation is comparatively less as apache server, database system and a webcam are needed so it is cost efficient.

#### Operational Feasibility:

The system makes a direct connection between the user and the check-in system therefore no man power is needed. So operational feasibility is achieved.

#### Social & Environmental Feasibility:

After the spread of this global pandemic this project will help to maintain all the social distancing guidelines makes the system gesture based instead of touch based.

### 3.3 Timeline Chart

TIMELINE CHART FOR AIRPORT CHECK IN HAND GESTURES												
MONTHS	FEBRUARY				MARCH				APRIL			
WEEK WORK	1	2	3	4	1	2	3	4	1	2	3	4
REQUIREMENT GATHERING												
UNDERSTANDING THE SUBJECT												
PROJECT PLANNING												
DESIGNING UI												
FINDING AND DEVELOPING MODEL												
TRAINING AND TESTING GESTURES												
TESTING THE FINAL PROJECT												
PROJECT REPORT												

FIG 3.2 – Timeline Chart

## **CHAPTER 4**

### **REQUIREMENT ANALYSIS**

## 4. Requirement Analysis

### 4.1 HARDWARE REQUIREMENTS

- Processor: Intel i7 (9<sup>th</sup> Gen)
- RAM: 8 GB or above
- Hard Disk: 1 TB or more
- GPU: NVIDIA GTX 1050Ti
- Webcam

### 4.2 SOFTWARE REQUIREMENTS

- Windows OS
- VS code
- Tensorflow.js
- JavaScript
- HTML
- CSS
- React
- Electron
- Mediapipe hands model
- Handpose model

### 4.3 FUNCTIONAL REQUIREMENT

In software engineering, a functional requirement defines a function of software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. As defined in requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

### 4.4 NON-FUNCTIONAL REQUIREMENT

In system engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of system, rather than specific behaviors. This should be contrasted with functional requirements is detailed in the system design. The plan for implementing non-functional requirements in detailed in the system architecture.

In general, functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Functional requirements are in the form of system shall do, while non-functional requirements are in the form of system shall be.

## 4.5 DATA FLOW DIAGRAM

### LEVEL 0 DFD



FIG 4.1 – DFD level 0

### LEVEL 1 DFD

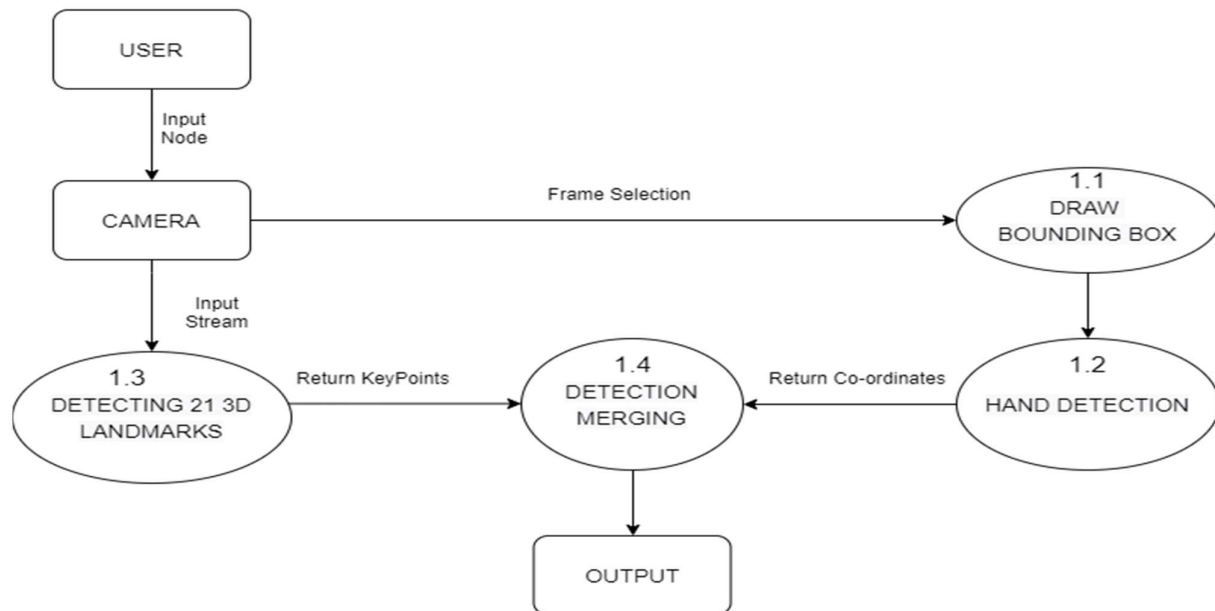


FIG 4.2 – DFD level 1

## 4.6 PROJECT METHODOLOGY

The UI is made using HTML CSS and JavaScript and embedded in electron app so that the web application can be used outside the browser as an windows application. The UI is made very simple and user friendly so that any user can use it without any prior knowledge.

The handpose model is used to detect the hand and recognize the gestures using a webcam. The tensorflow handpose model is used to detect 21 3D landmarks of a hand from a live video stream. The handpose model is based on google mediapipe hands which is embedded in tensordlow.js which can be used in web applications. The landmarks to be used are emphasized which can then be used to detect click function when the thumb tip and index finger tip is pinched twice. The centroid is used to place a cursor on the UI.

First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for

two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects. Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance.

When the hand is moved in front of the webcam the hand is detected using the hand detection model and the UI can be navigated using a mouse like simulated gestures which can be used to use the check-in system.

## **CHAPTER 5**

### **PROPOSED SYSTEM**



## 5. PROPOSED SYSTEM

### 5.1 SYSTEM DESIGN

#### 5.1.1 FLOW DIAGRAM

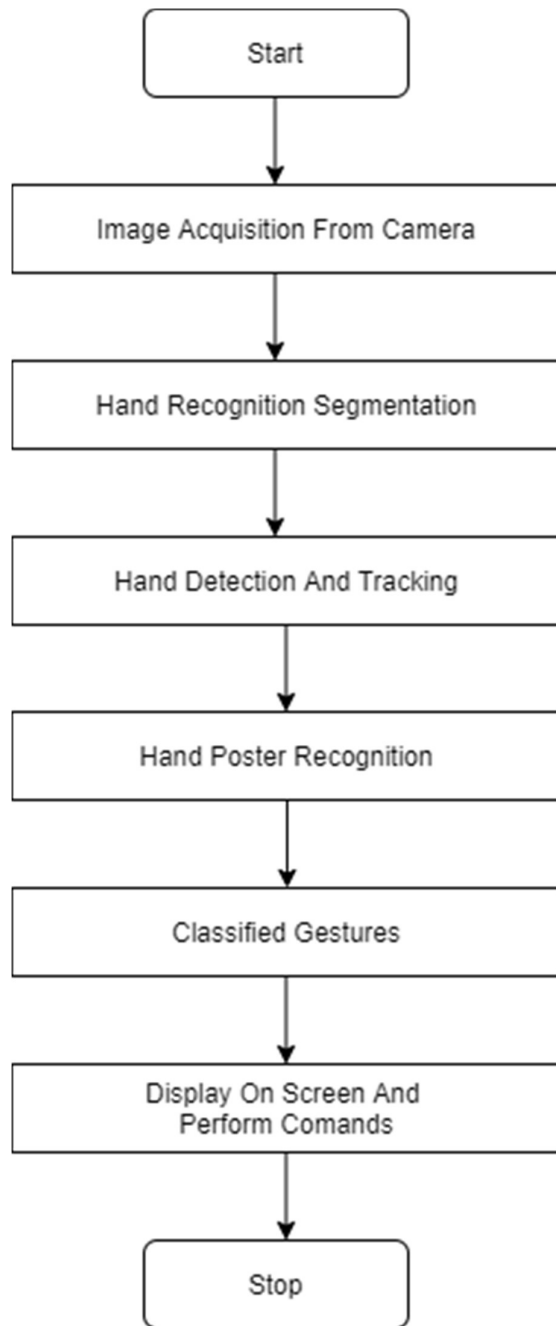


FIG 5.1- Flow Chart

### 5.1.2 USE CASE DIAGRAM

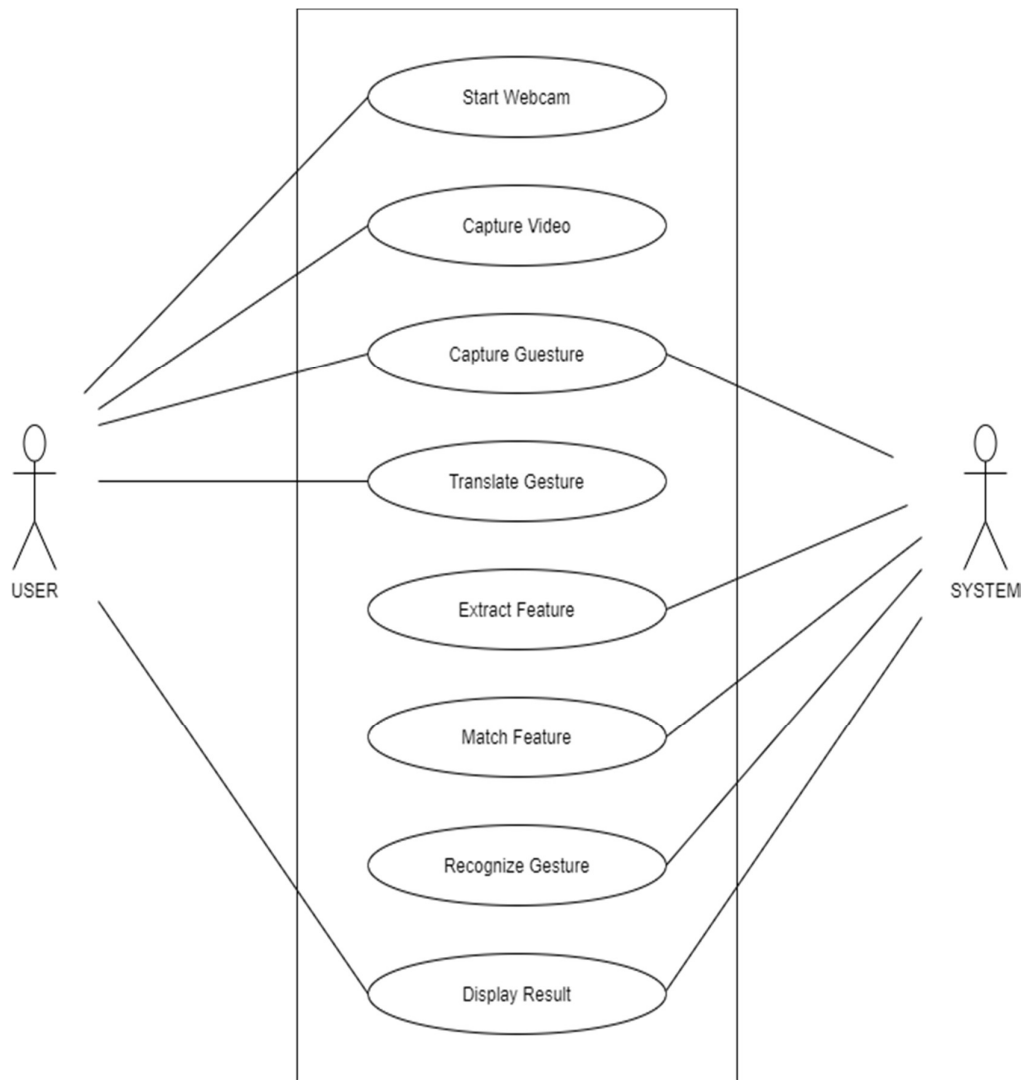


FIG NO. 5.2 – Use case diagram

### 5.1.3 ACTIVITY DIAGRAM

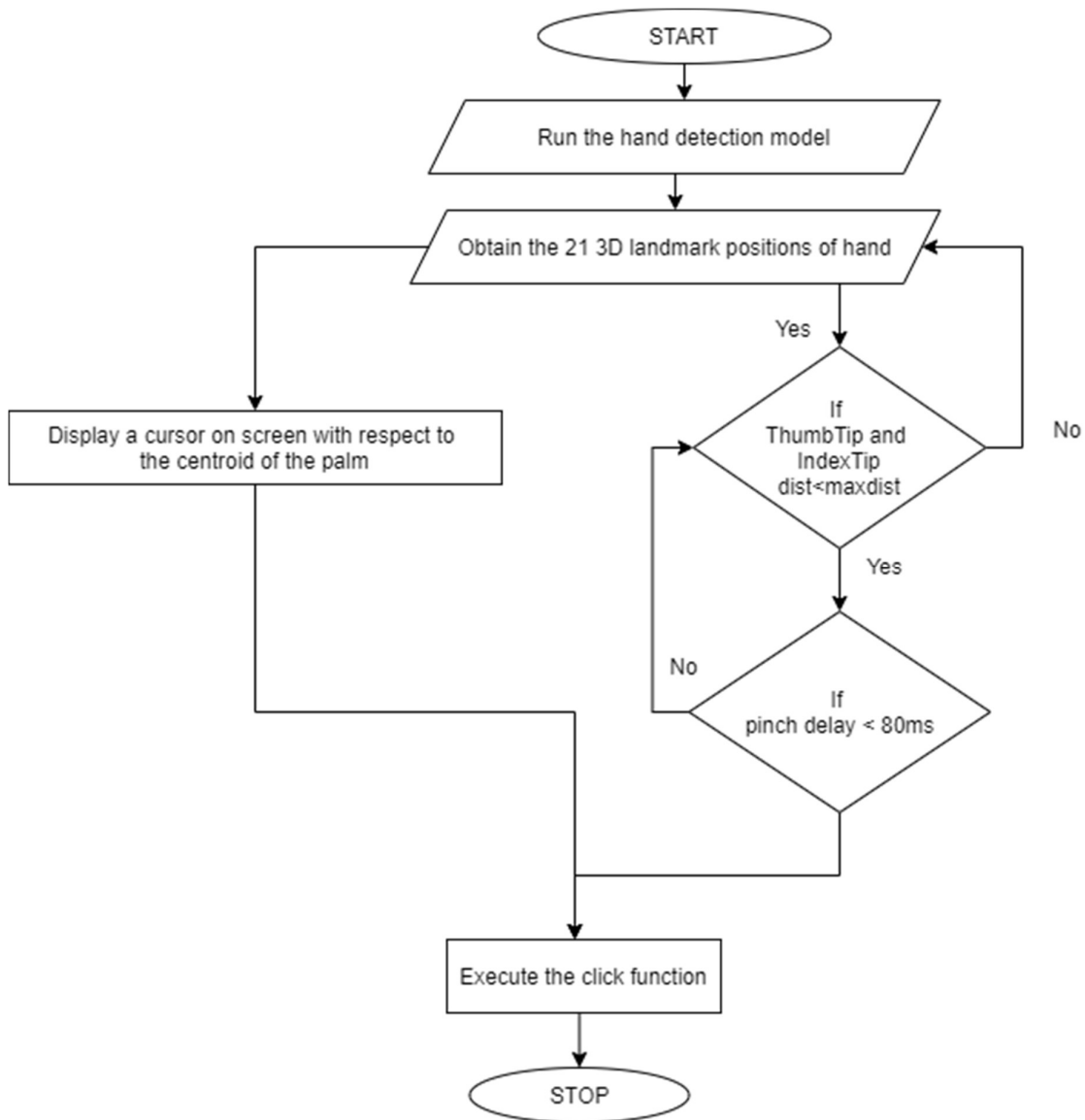


FIG NO. 5.3 – Activity Diagram

### 5.1.4 SEQUENCE DIAGRAM

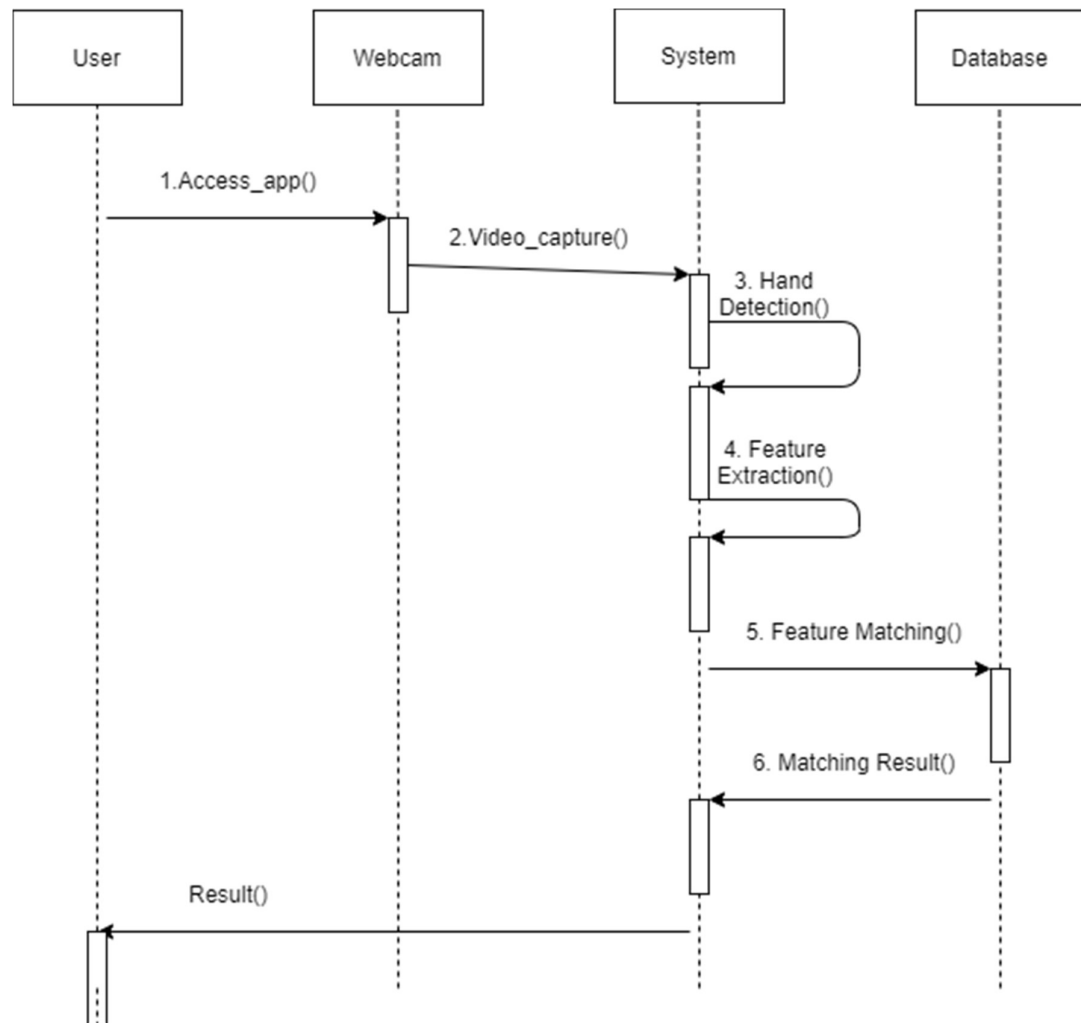


Fig No 5.4 – Sequence Diagram

### 5.1.5 CLASS DIAGRAM

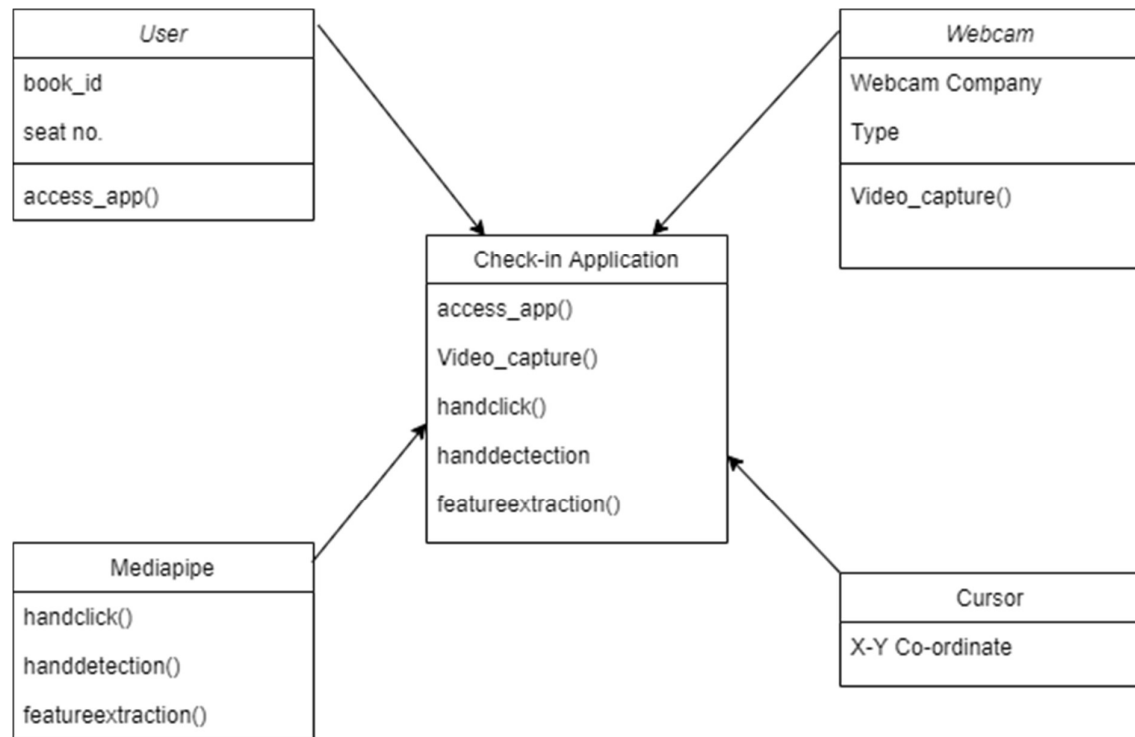


FIG 5.5 – Class Diagram

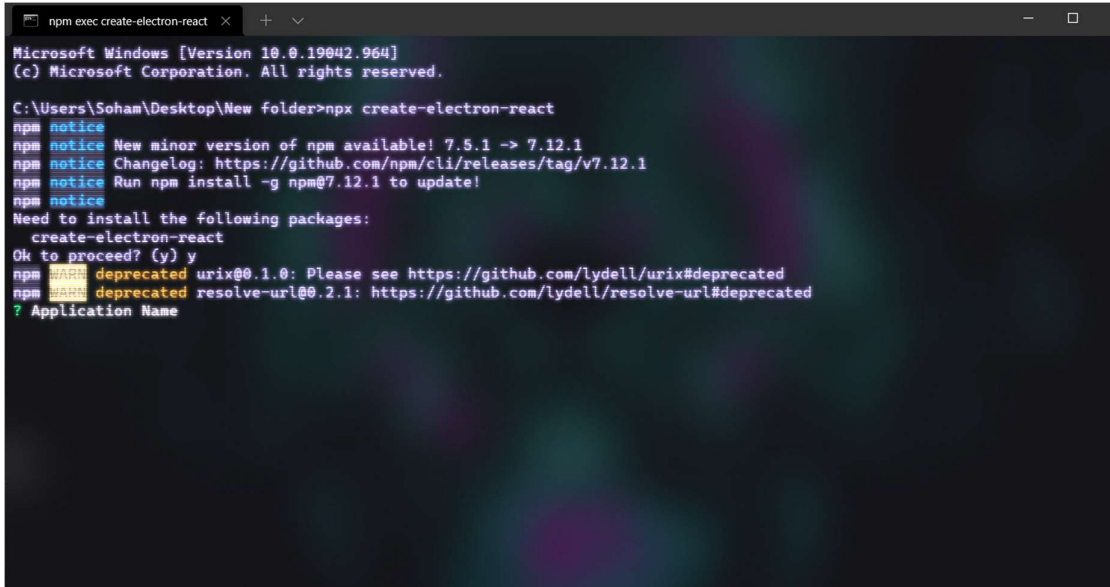
## **CHAPTER 6**

### **IMPLEMENTATION**

## 6. IMPLEMENTATION

### 6.1 CREATING UI

#### STEP 1: Create Electron React App



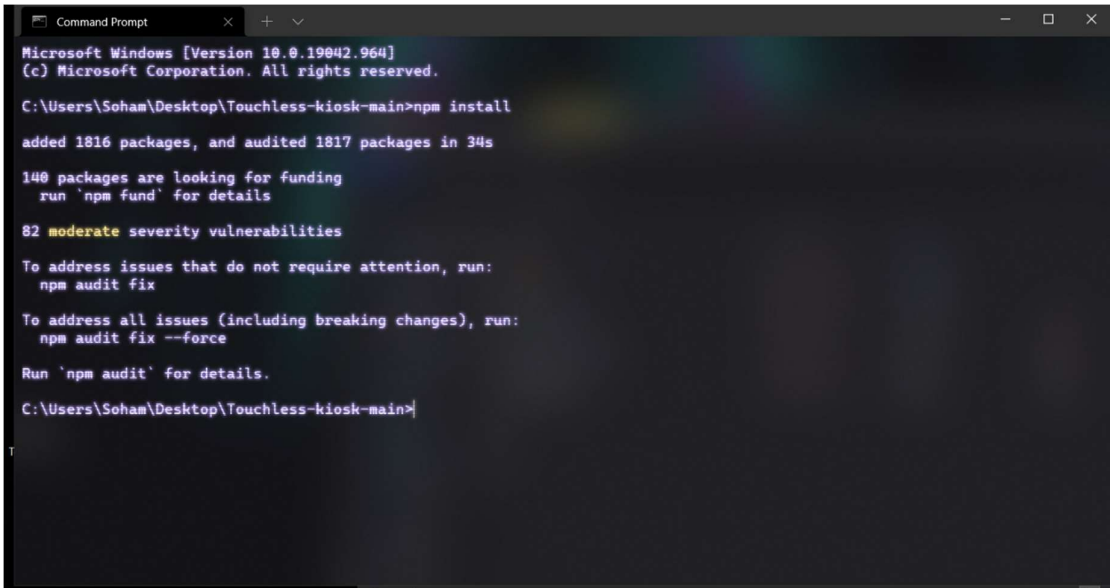
```

Microsoft Windows [Version 10.0.19042.964]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Soham\Desktop\New folder>npx create-electron-react
npm notice
npm notice New minor version of npm available! 7.5.1 -> 7.12.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.12.1
npm notice Run npm install -g npm@7.12.1 to update!
npm notice
Need to install the following packages:
  create-electron-react
Ok to proceed? (y) y
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
? Application Name
  
```

FIG 6.1 – Creating electron app

#### STEP 2: Installing required node modules



```

Microsoft Windows [Version 10.0.19042.964]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Soham\Desktop\Touchless-kiosk-main>npm install

added 1816 packages, and audited 1817 packages in 34s

140 packages are looking for funding
  run 'npm fund' for details

82 moderate severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

C:\Users\Soham\Desktop\Touchless-kiosk-main>
  
```

FIG 6.2 – Installing packages

**STEP 3:** Designing the frontend using HTML, CSS and JavaScript

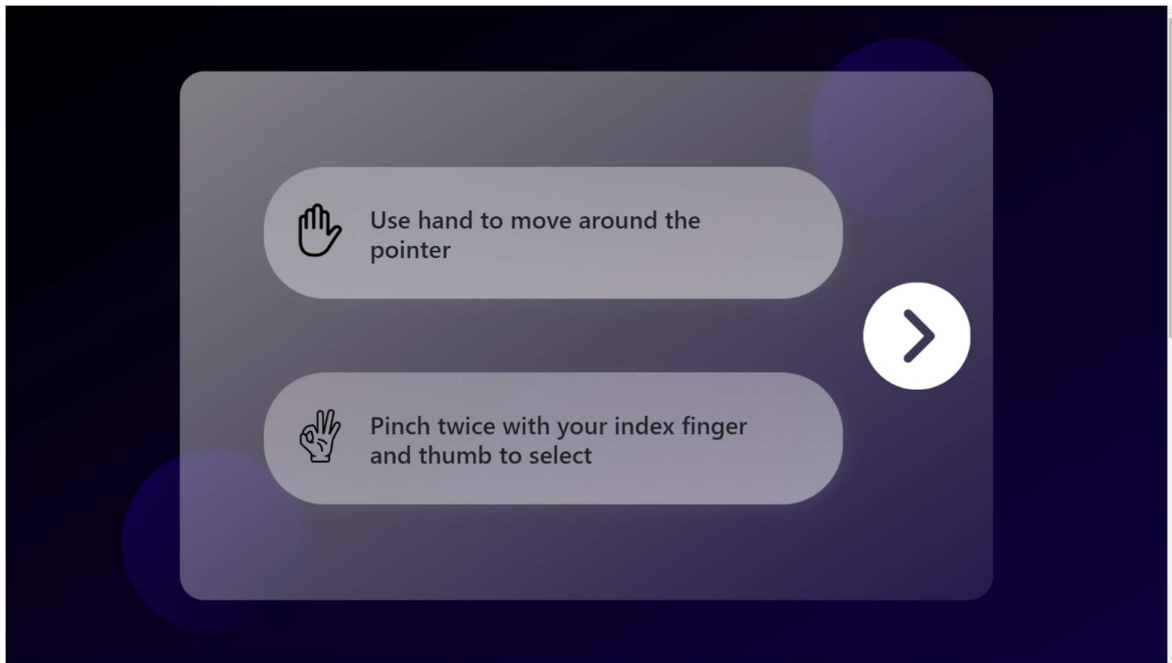


FIG 6.3 – UI – How to use the gestures

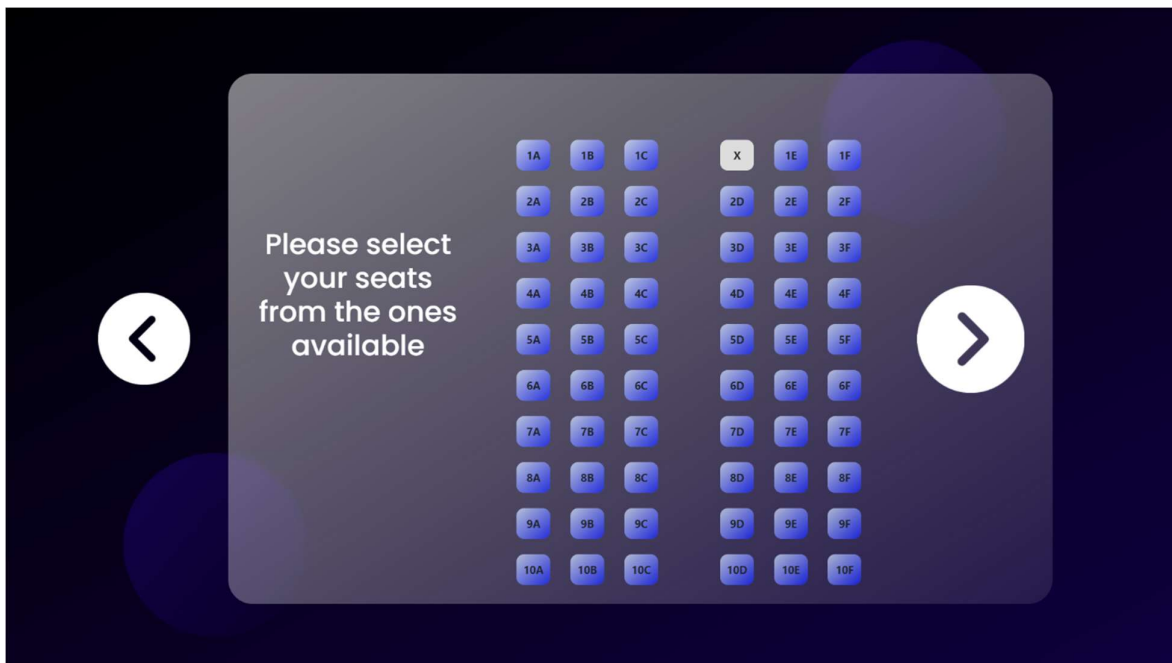


FIG 6.4 – UI – Seat selection



## 6.2 Selection of hand detection model

Tensorflow Hand pose model:

The Tensorflow hand detection model is combination of Google's MediaPipe Hands model and tensorflow.js

The handpose package detects hands in an input image or video stream, and returns twenty-one 3-dimensional landmarks locating features within each hand. Such landmarks include the locations of each finger joint and the palm.

```
// Load the MediaPipe handpose model assets.
const model = await handpose.load();

// Pass in a video stream to the model to obtain
// a prediction from the MediaPipe graph.
const video = document.querySelector("video");
const hands = await model.estimateHands(video);

// Each hand object contains a `landmarks` property,
// which is an array of 21 3-D landmarks.
hands.forEach(hand => console.log(hand.landmarks));
```

FIG 6.5 – Loading model

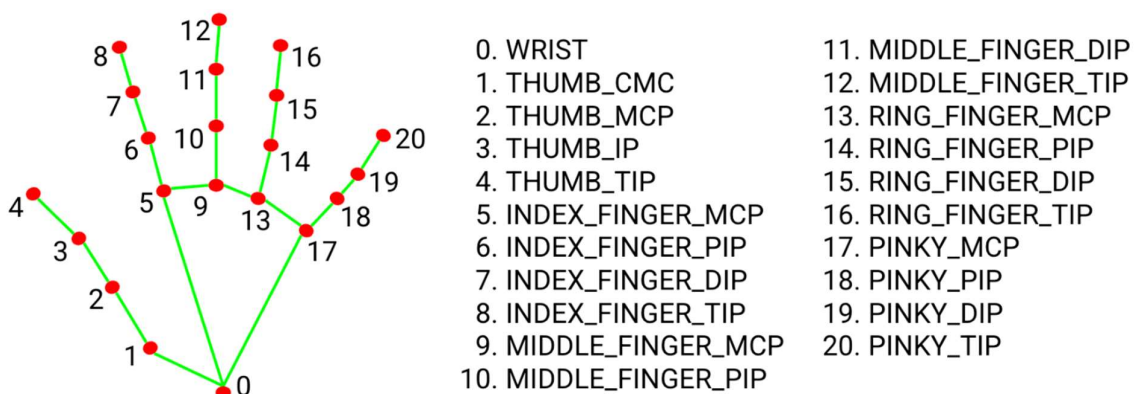


FIG 6.6 – 21 3D landmarks plotted on hand by Mediapipe

### 6.3 IMPLEMENTING THE MODEL

#### 1) Finding the centroid of palm:

The centroid of the palm is used as a point which is then further used to measure coordinates of that point with respect to the screen to plot a cursor which moves according to the movement of hand in front of the camera. The point is then marked visible on the screen which can be used to represent a mouse pointer.

```
if ($el) {
  $el.dispatchEvent(
    new MouseEvent(event, {
      view: window,
      button: 0,
      bubbles: true,
      cancelable: true,
      clientX: pointer.x,
      clientY: pointer.y,
      // Only used when the mouse is captured in full screen mode
      movementX: pointer.x - lastHeld[hand].x,
      movementY: pointer.y - lastHeld[hand].y
    })
  );
}
```

FIG 6.7 – Selecting and displaying the centroid of hand

#### 2) Plotting hand points:

The 21 3D landmarks of the hand are plotted which represent different features of the hand. Thumb\_Tip and Index\_Finger\_Tip points are then further used to emphasize as click points to be used as click gesture.

```
class HandposeModel extends BaseModel {
  constructor (handsfree, config) {
    super(handsfree, config);
    this.name = 'handpose';

    // Various THREE variables
    this.three = {
      scene: null,
      camera: null,
      renderer: null,
      meshes: []
    };

    this.normalized = [];

    // landmark indices that represent the palm
    // 8 = Index finger tip
    // 12 = Middle finger tip
    this.palmPoints = [0, 1, 2, 5, 9, 13, 17];
    this.gestureEstimator = new fingerpose$1.GestureEstimator([]);
  }
}
```

FIG 6.8 – Plotting landmarks of hand

## 6.4 TRAINING MODEL

In this step we train model for 400-500 iterations where in with every iteration the model reduces its error in predictions and increases the accuracy by recording subtle variations in hand points and 3D landmarks. The Handsfree() is used in order to record the hand variations and capture and save their data which can be then used to create new gesture.



FIG 6.9 – Training the model for pinch gesture

## 6.5 SETTING CLICK EVENTS:

In this step we enable the mouse click events to simulate the cursor using the trained model and the handpose keypoints. The throttle (Time delay between gestures) is set to 80 milliseconds to improve the accuracy of the click and to avoid getting random double clicks which may cause errors or unusual behavior while using the system.

```
var pluginHandClick = {
  models: 'Hands',
  enabled: true,

  tags: ['browser'],

  config: {
    // How often in milliseconds to trigger clicks
    throttle: 80,

    // Max number of frames to keep down
    maxMouseDownFrames: 1,

    // Morphs to watch for and their required confidences
    morphs: {
      0: 0.25,
      1: 0.25
    }
  },

  // Number of frames mouse has been downed
  mouseDowned: 0,
  // Is the mouse up?
  mouseUp: false,
  // Whether one of the morph confidences have been met
  thresholdMet: false,

  // The last held {x, y}, used to calculate move delta
  lastHeld: {x: 0, y: 0},
  // Original target under mousedown
  $origTarget: null,
```

FIG 6.10 – Setting the click events

## 6.6 ALGORITHM:

The tensorflow handpose model is used to detect 21 3D landmarks of a hand from a live video stream. The handpose model is based on google mediapipe hands which is embedded in tensordlow.js which can be used in web applications.

For the gestures to work:

- 1) Given an input, the model predicts whether it contains a hand. If so, the model returns coordinates for the bounding box around the hand, as well as 21 keypoints within the hand, outlining the location of each finger joint and the palm.
- 2) The centroid of hand is then taken which is then used to simulate the cursor of the mouse. As the centroid moves along the screen the x and y distance is taken and the cursor is placed on the screen which is visible to the user and can be used to navigate around in the UI.
- 3) The gesture is trained for the click function using the Handsfree() module where the hand variations are captured and stored to work where the Thumb\_Tip and Index\_Finger\_Tip is pinched twice which can be then implemented as a new gesture in the web application.
- 4) The click function is implemented with the cursor which then is used together to simulate a mouse where the click is performed when the index finger and thumb is pinched twice within a duration of 80ms.

## **CHAPTER 7**

### **RESULT**

## 7. RESULT

### DETECTION OF HAND AND 21 3D POINTS

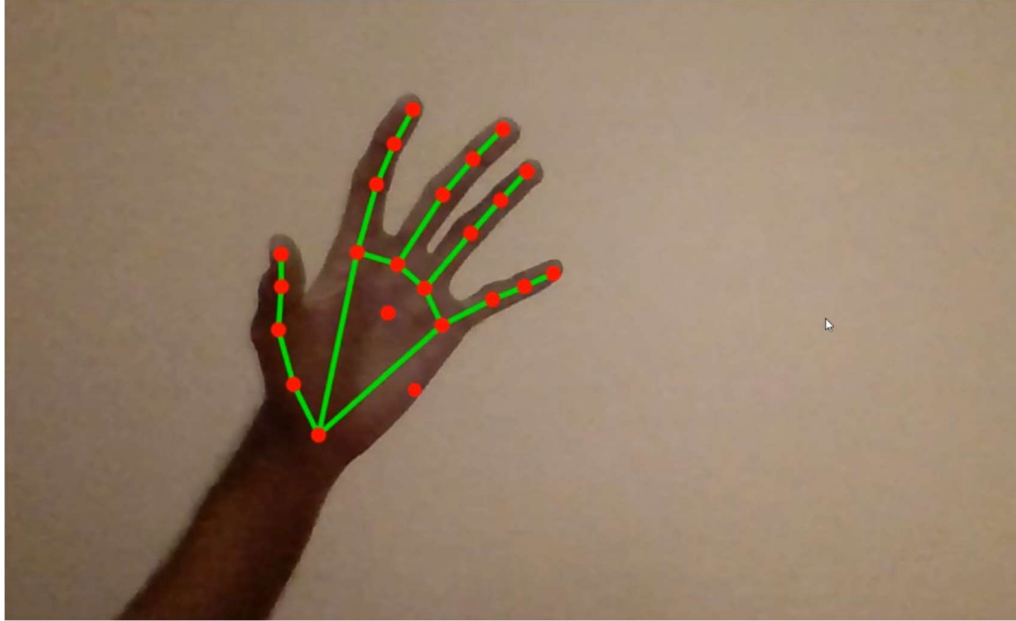


FIG 7.1 – Hand detection with 21 3D land marks

### TYPING AND NAVIGATION USING HAND GESTURES

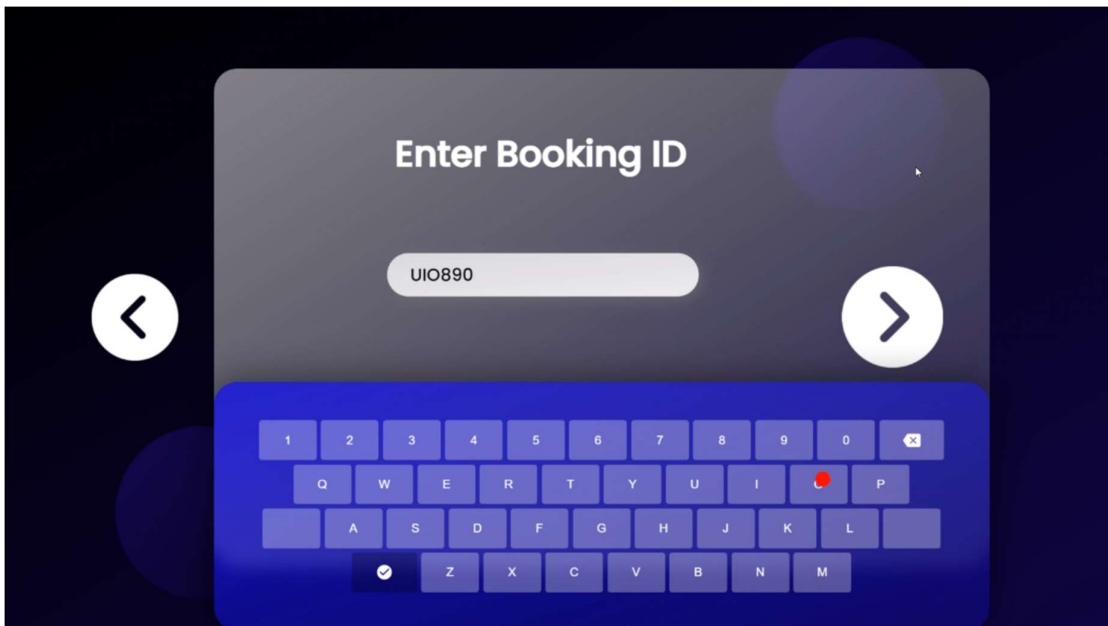


FIG 7.2 – Typing using the hand gestures

## **CHAPTER 8**

## **CONCLUSION**



## **8. CONCLUSION**

### **8.1 CONCLUSION**

Touch-based systems aren't safe anymore, as it can contribute in the spread of disease. This Touch less check-in kiosk ensures that it is safe to use as compared to traditional kiosk. With the help of hand gestures, user successfully collect boarding pass. This touch less hand gesture-based model is very accurate and capable of performing all the steps with the help of Tensor flow hand pose model. Thus, using deep learning models and the very user friendly UI we have successfully achieved our objectives and implemented gestures to navigate and use the Check-in system.

### **8.2 FUTURE SCOPE**

The project will have the following upcoming features-

1. Scanning Passport to retrieve the booking details.
2. Improving accuracy of the model and reduce load times.

## REFERENCES

- ❖ <https://towardsdatascience.com/handtrackjs-677c29c1d585>
- ❖ <https://sci-hub.do/https://ieeexplore.ieee.org/document/8777621>
- ❖ <https://google.github.io/mediapipe/solutions/hands>
- ❖ <https://blog.tensorflow.org/2020/03/face-and-hand-tracking-in-browser-with-mediapipe-and-tensorflowjs.html>
- ❖ <https://www.electronjs.org/docs/tutorial/quick-start#electron-api>

## ACKNOWLEDGEMENT

It is pleasant task to express gratitude to all those who contributed in many ways to this project and made it an unforgettable experience for us. First of all, we would like to thank our guide Prof. Deepti Vijay Chandran. This work would not have been possible without her guidance, support and encouragement.

We are highly indebted to Dr. Sunil Chavan, Principal, **Smt. Indira Gandhi College of Engineering**, Prof. Sonali Deshpande, Head of the Department, **Computer Department** and also all the faculty members.

We are thankful to lord for keeping us energized that every end seems to be new beginning.

Airport Check-in system using hand gestures

Airport Check-in system using hand gestures

Airport Check-in system using hand gestures

Airport Check-in system using hand gestures

