Deployment manual

Fact Finder

accuracy, integrity and fairness

FactFinder
May 2024
Team 4

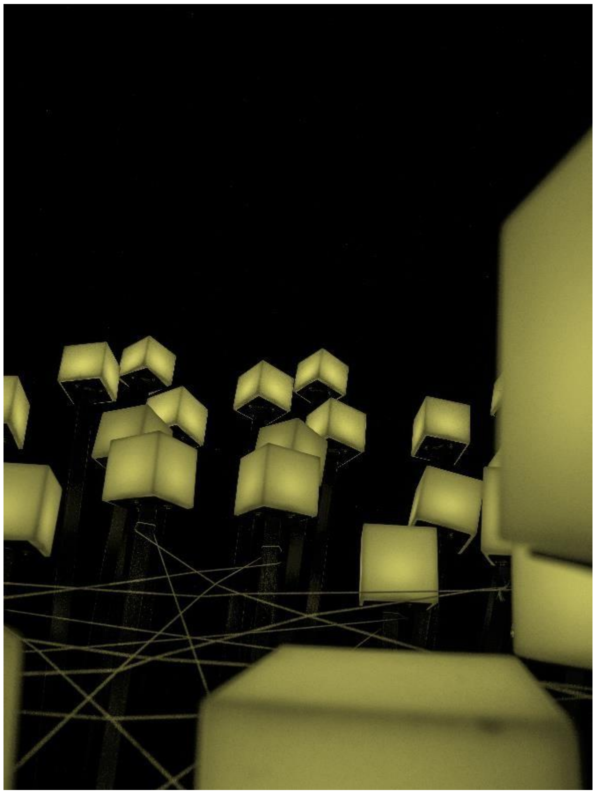
Table of Contents

Introduction

Hello everyone, welcome to the deployment manual for “FactFinder” application. This document is designed to provide a complete step by step solution for successful deployment of our software or webiste at your system/environment. It is meant for use by developers, advanced users, system administrators or simple end users.

Following topics will cover in this manual:

1. **System Requirements** – part where we outline hardware/software requirements.
2. **Installation procedures** – step by step guide into how to install software in your system.
3. **Troubleshooting** – outline most common errors and issues that might arise.
4. **Support and resources** – we will provide links to related resources to understand our stack better.



System Requirements

Our software will run on major system operating systems without having a big impact on hardware components. Before attempting to install software components please make sure your system meets the following hardware requirements. Furthermore, we have reduced our requirement for mobile devices and desktop computers.

Hardware requirements (Desktop) Processor

(CPU):
Equivalent to AMD Ryzen 3 2.4Ghz (4-core)/ Intel Core i-3 2.6Ghz(4-core physical)/ Apple M1 (8-core)

Memory (RAM):
A minimum of 8 Gb DDR4 @ 4200 Mhz

Storage:
5 Gb of SSD space or equivalent for optimal performance

GPU:
Intel Iris Xe with 1.30 Ghz/ Radeon Graphics 2.00 Ghz

Screen Resolution:
1920 x 1080

Network Connectivity:
Minimum download speed of 1Mpbs

Peripheral Devices:
Camera with at least 5MP lens and autofocus.

Software requirements (Desktop)

Operating System:
Windows 8/10/11
MacOS Monterey/Big Sur/Catalina/Mojave Ubuntu
Jammy Jellyfish/Mantic Minotaur

System Requirements

Hardware requirements (Mobile) **Processor**

(CPU):
Equivalent to Mediatek MT6765 Helio P35 (12nm)/ A15 Bionic

Memory (RAM):
A minimum of 2 Gb

Storage:
800 Mb

GPU:
PowerVR GE8320

Screen Resolution:
720 x 1334

Network Connectivity:
Minimum download speed of 1Mpbs

Peripheral Devices:
Front Facing Camera with at least 5MP lens and autofocus.

Software requirements (Mobile)

Operating System:
Apple iOS 15/16/17
Android 11/12/13/14

Installation Procedures

Installing Visual Studio Code (VS Code):

1. **Download VS Code:**
   - Visit the official Visual Studio Code website at https://code.visualstudio.com/.
   - Click on the "Download" button to download the installer for your operating system (e.g., Windows, macOS, or Linux).
2. **Install VS Code:**
   - Once the installer is downloaded, run the installer executable.
   - Follow the on-screen instructions to install VS Code, including accepting the license agreement.
   - Choose your preferred installation settings and location (you can usually leave the default settings as they are).
3. **Launch VS Code:**
   - After installation, you can launch VS Code from your system's application menu or desktop shortcut.

Installing Node.js and npm:

1. **Download Node.js:**
   - Open your web browser and visit the official Node.js website at https://nodejs.org/.
   - On the website, you will see two versions: "LTS" (Long-Term Support) and "Current." It's generally recommended to install the LTS version for stability.
   - Click on the "LTS" or "Current" version, depending on your preference.
2. **Install Node.js:**
   - Once the installer is downloaded, run the installer executable.
   - Follow the installation wizard's instructions. You can typically use the default settings for a typical installation.
   - During the installation process, you may be asked to accept the terms and conditions, so be sure to review and agree if prompted.
3. **Verify Node.js and npm Installation:**
   - Open a command prompt or terminal window.
   - To verify that Node.js and npm have been successfully installed, type the following commands and press Enter:

   - node -v
   - npm -v

- You should see the installed Node.js and npm versions displayed in the terminal.

That's it! You've now successfully installed Visual Studio Code and Node.js with npm. You're ready to start using these tools for your development projects.

Cloning Git Repository

Open a Terminal or Command Prompt:

- On your local computer, open a terminal (Linux/macOS) or a command prompt (Windows). You'll use this to run Git commands.

Navigate to the Directory Where You Want to Clone the Repository:

- Use the cd (change directory) command to navigate to the location where you want to clone the repository. For example, to clone the repository in your home directory.

Clone the Repository:

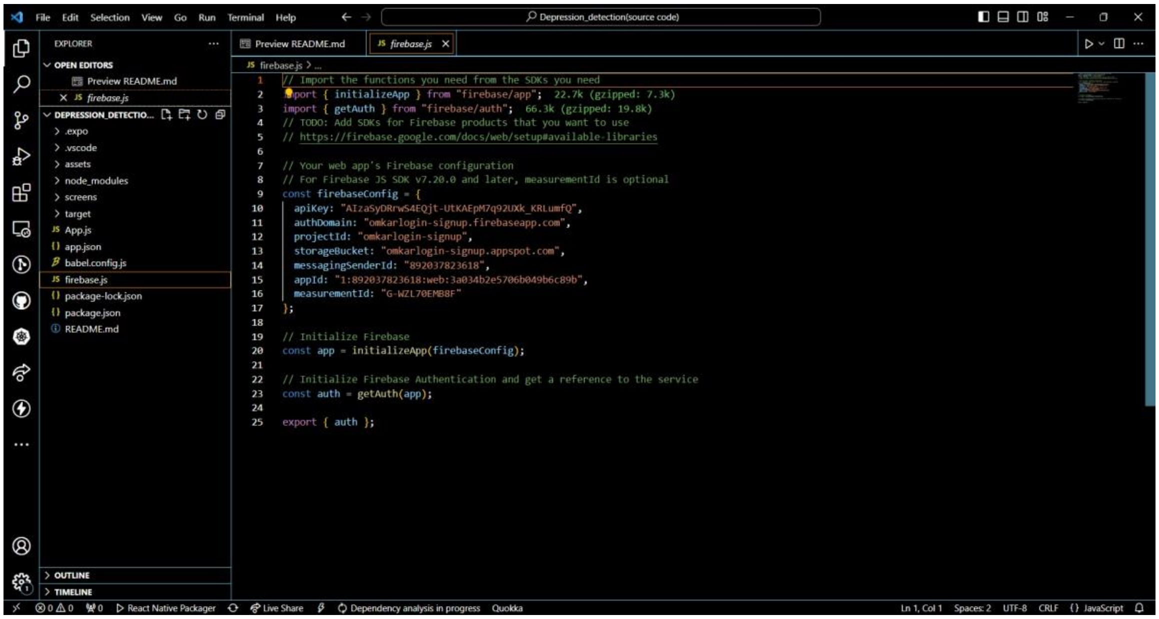
- Use the git clone command followed by the GitHub repository. GitHub repository command will be:

```
git clone https://github.com/htmw/2023S-Team2
```

Check the Cloned Repository:

- After the cloning process is complete, you'll have a local copy of the repository in a directory with the same name as the repository (e.g., "2023S-Team2"). You can navigate into this directory using the cd command.

Now, you have successfully cloned the Git repository from the provided GitHub URL. You can work with the repository's files and use Git to manage version control, such as making commits, branching, and pushing changes back to the remote repository on GitHub.



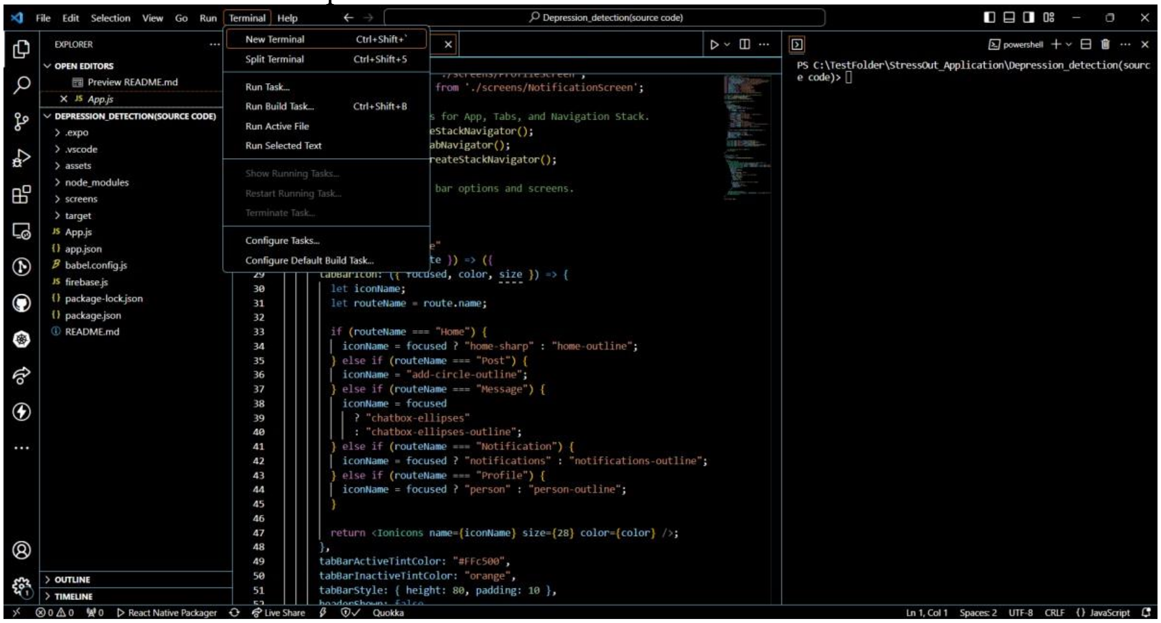
```javascript
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";  22.7k (gzipped: 7.3k)
import { getAuth } from "firebase/auth";  66.3k (gzipped: 19.8k)
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDRrwS4EQjt-UtKAEpM7q92UXk_KRLumfQ",
  authDomain: "omkarlogin-signup.firebaseapp.com",
  projectId: "omkarlogin-signup",
  storageBucket: "omkarlogin-signup.appspot.com",
  messagingSenderId: "892037823618",
  appId: "1:892037823618:web:3a034b2e5706b049b6c89b",
  measurementId: "G-WZL70EMB8F"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);

// Initialize Firebase Authentication and get a reference to the service
const auth = getAuth(app);

export { auth };
```

After successfully cloning git reporsitory your VS code must look similar to the screenshot above.

Starting the server

Creating new terminal instance:

In VS Code we need to open a new terminal: terminal → new terminal

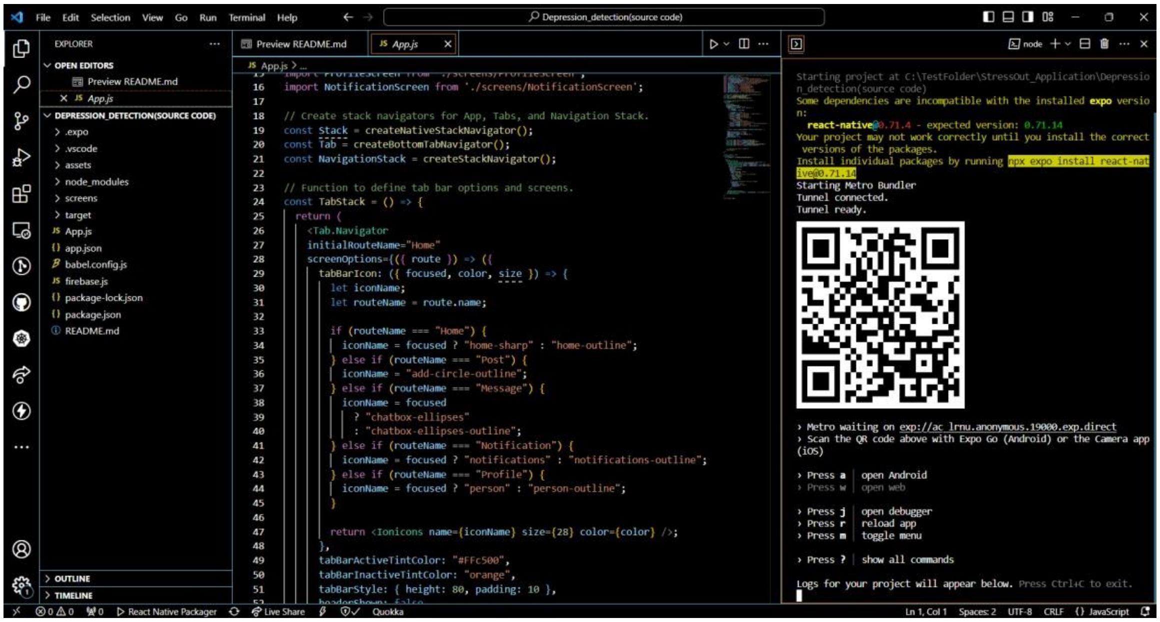


```
 from './screens/NotificationScreen';

s for App, Tabs, and Navigation Stack.
eStackNavigator();
abNavigator();
reateStackNavigator();

 bar options and screens.

e"
te }) => ({
tabBarIcon: ({ focused, color, size }) => {
  let iconName;
  let routeName = route.name;

  if (routeName === "Home") {
    iconName = focused ? "home-sharp" : "home-outline";
  } else if (routeName === "Post") {
    iconName = "add-circle-outline";
  } else if (routeName === "Message") {
    iconName = focused
      ? "chatbox-ellipses"
      : "chatbox-ellipses-outline";
  } else if (routeName === "Notification") {
    iconName = focused ? "notifications" : "notifications-outline";
  } else if (routeName === "Profile") {
    iconName = focused ? "person" : "person-outline";
  }

  return <Ionicons name={iconName} size={28} color={color} />;
},
tabBarActiveTintColor: "#FFc500",
tabBarInactiveTintColor: "orange",
tabBarStyle: { height: 80, padding: 10 },
```

```
PS C:\TestFolder\StressOut_Application\Depression_detection(sourc
e code)>
```

Running the server:

Run npm startto start the server.



```
import ProfileScreen from './screens/ProfileScreen';
import NotificationScreen from './screens/NotificationScreen';

// Create stack navigators for App, Tabs, and Navigation Stack.
const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();
const NavigationStack = createStackNavigator();

// Function to define tab bar options and screens.
const TabStack = () => {
  return (
    <Tab.Navigator
    initialRouteName="Home"
    screenOptions={({ route }) => ({
      tabBarIcon: ({ focused, color, size }) => {
        let iconName;
        let routeName = route.name;

        if (routeName === "Home") {
          iconName = focused ? "home-sharp" : "home-outline";
        } else if (routeName === "Post") {
          iconName = "add-circle-outline";
        } else if (routeName === "Message") {
          iconName = focused
            ? "chatbox-ellipses"
            : "chatbox-ellipses-outline";
        } else if (routeName === "Notification") {
          iconName = focused ? "notifications" : "notifications-outline";
        } else if (routeName === "Profile") {
          iconName = focused ? "person" : "person-outline";
        }

        return <Ionicons name={iconName} size={28} color={color} />;
      },
      tabBarActiveTintColor: "#FFc500",
      tabBarInactiveTintColor: "orange",
      tabBarStyle: { height: 80, padding: 10 },
      headerShown: false
```

```
Starting project at C:\TestFolder\StressOut_Application\Depressio
n_detection(source code)
Some dependencies are incompatible with the installed expo versio
n:
  react-native@0.71.4 - expected version: 0.71.14
Your project may not work correctly until you install the correct
 versions of the packages.
Install individual packages by running npx expo install react-nat
ive@0.71.14
Starting Metro Bundler
Tunnel connected.
Tunnel ready.

› Metro waiting on exp://ac lrnu.anonymous.19000.exp.direct
› Scan the QR code above with Expo Go (Android) or the Camera app
(iOS)

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu

› Press ? │ show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
```

Scan the QR-code to start using the application. You can commands written in the terminal to control launched application.

› Scan the QR code above with Expo Go (Android) or the Camera app
(iOS)

› Press a │ open Android
› Press w │ open web

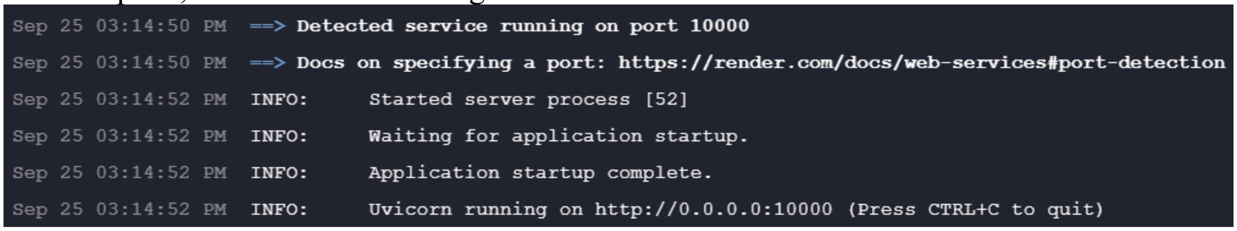
› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu

› Press ? │ show all commands

Machine Learning API deployment

Creating new web service:

- Go to https://dashboard.render.com/
- On top right, navigate to New+ → web services
- Build and deploy from a Git repository → Next
- Connect to the repository where the detection API is located at (Clone from https://github.com/YuxiangLiuGC/Detection_API and upload on your own GitHub repository)
- Under the deployment page, name the app and select the region of your preference
- Make sure the Environment to be "Python 3"
- Build Command needs to be "pip install -r requirement.txt"
- Start Command needs to be "uvicorn model_API:app –host 0.0.0.0 –port 10000"
- Select the instance type from different hardware specifications
- When finished, click Create Web Service
- Under the dashboard page, go to Logs to see the deployment process
- When you see your terminal showing logs similar to this, the deployment process is complete, and the API is running



```
Sep 25 03:14:50 PM  ==> Detected service running on port 10000
Sep 25 03:14:50 PM  ==> Docs on specifying a port: https://render.com/docs/web-services#port-detection
Sep 25 03:14:52 PM  INFO:     Started server process [52]
Sep 25 03:14:52 PM  INFO:     Waiting for application startup.
Sep 25 03:14:52 PM  INFO:     Application startup complete.
Sep 25 03:14:52 PM  INFO:     Uvicorn running on http://0.0.0.0:10000 (Press CTRL+C to quit)
```

Feedback and version information

Date: 06th of November 2023
Version 0.9

Feedback and Updates

We value your feedback and are committed to continuously improving the deployment process and the software itself. Your input is invaluable to us, and we encourage you to share your thoughts, suggestions, or report any issues you encounter during the deployment.

Email: **ak71778n@pace.edu**