LAMPIRAN

Lampiran 1: Uraian Kegiatan

Berikut merupakan tabel uraian kegiatan yang kami lakukan selama masa PKL (Praktek Kerja Lapangan), yang mengangkat judul "PEMBUATAN WEBSITE PORTAL INFORMASI STUDYINJOGJA.COM PRAKTEK KERJA LAPANGAN"

Minggu	Tanggal	Kegiatan
Minggu 1	10 - 13 Januari	• Pengembangan Kuliner
Minggu 2	16 - 20 Januari	• Pengembangan Kuliner
Minggu 3	23 - 27 Januari	• Pengembangan Study
Minggu 4	30 Januari - 3 Februari	• Pengembangan Wisata

Lampiran 2: Source Code

Berikut merupakan Source Code Pembukaan Account Admin dan Booking

Fitur	Source Code
Register	`controller.register = async (req, res) => {` `  try {` `  const salt = await bcrypt.genSalt(10);` `  const hash = await bcrypt.hash(req.body.password, salt);` `  await model.user` `  .findOne({` `  where: {` `  email: req.body.email,` `  },` `  })` `  .then((result) => {` `  if (result) {` `  res.status(400).json({` `  message: "email sudah terdaftar",` `  status: 400,` `  });` `  } else {`

	`` model.user .create({ nama: req.body.nama, email: req.body.email, uid: uid(), password: hash, role: "user", createdAt: new Date().getTime(), updatedAt: new Date().getTime(), }) .then((result) => { res.status(200).json({ message: "register berhasil", status: 200, data: result, }); }); } }); } catch (error) { res.status(400).json({ message: error.message, status: 400, data: [], }); } }; ```
Login	``` controller.login = async (req, res) => { try { await model.user .findOne({ where: { email: req.body.email, }, }) .then(async (result) => { if (result) { const validPassword = await bcrypt.compare(req.body.password, result.password ```

```
          );
          if (validPassword) {
            const token = jwt.sign(
              {
                id: result.id,
                uid: result.uid,
                nama: result.nama,
                email: result.email,
                role: result.role,
              },
              SECRET_KEY,
              {
                expiresIn: "24h",
              }
            );
            res.status(200).json({
              message: "login berhasil",
              status: 200,
              data: result,
              token: token,
            });
          } else {
            res.status(400).json({
              message: "password salah",
              status: 400,
            });
          }
        } else {
          res.status(404).json({
            message: "email tidak terdaftar",
            status: 400,
          });
        }
      });
  } catch (error) {
    res.status(400).json({
      message: error.message,
      status: 400,
      data: [],
    });
  }
};
```

Update Account	``` controller.update = async (req, res) => { const data = { nama: req.body.nama, email: req.body.email, password: req.body.password, updatedAt: new Date().getTime(), role: req.body.role, }; try { if (req.body.password) { const salt = await bcrypt.genSalt(10); const hash = await bcrypt.hash(req.body.password, salt); data.password = hash; } await model.user .findOne({ where: { id: req.params.id, }, }) .then((result) => { if (result) { if (data.email) { model.user .findOne({ where: { email: req.body.email, }, }) .then((result) => { if (result) { res.status(400).json({ message: "email sudah terdaftar", status: 400, }); } else { model.user .update(data, { where: { ```

```
                        id: req.params.id,
                      },
                    })
                    .then((result) => {
                      res.status(200).json({
                        message: "berhasil update
data",
                        status: 200,
                        data: result,
                      });
                    });
                }
              });
            } else {
              model.user
                .update(data, {
                  where: {
                    id: req.params.id,
                  },
                })
                .then((result) => {
                  res.status(200).json({
                    message: "berhasil update data",
                    status: 200,
                    data: result,
                  });
                });
            }
          } else {
            res.status(404).json({
              message: "data not found",
              status: 404,
              data: [],
            });
          }
        });
    } catch (error) {
      res.status(400).json({
        message: error.message,
        status: 400,
        data: [],
      });
```

	`` } };```
Delete Account	```controller.delete = async (req, res) => { try { await model.user .findOne({ where: { id: req.params.id, }, }) .then((result) => { if (result) { model.user .destroy({ where: { id: req.params.id, }, }) .then((result) => { res.status(200).json({ message: "berhasil hapus data", status: 200, data: result, }); }); } else { res.status(404).json({ message: "data not found", status: 404, data: [], }); } }); } catch (error) { res.status(400).json({ message: error.message, status: 400, data: [], }); } };```

Generate PDF	

```
controller.generatePdf = async (req, res) => {
  try {
    await model.booking
      .findOne({
        where: {
          uid: req.params.id,
        },
      })
      .then((result) => {
        if (result) {
          const doc = new PDFDocument();
          var date = new
Date(parseInt(result.jam_berkunjung));
          var hours = date.getHours();
          var minutes = "0" + date.getMinutes();
          var time = hours + ":" + minutes;

          doc.fontSize(18).text("Detail
Pemesanan", { align: "center" });
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc.fontSize(12).text("ID Pemesanan: ",
{
            continued: true,
            align: "left",
            width: 200,
          });
          doc.text(`${result.uid}`, { align:
"left" });
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc
            .fontSize(12)
            .text("Nama: ", { continued: true,
align: "left", width: 200 });
          doc.text(`${result.nama}`, { align:
"left" });
          doc.moveDown(); // Pindah ke baris
berikutnya
```

```
          doc
            .fontSize(12)
            .text("Email: ", { continued: true,
align: "left", width: 200 });
          doc.text(`${result.email}`, { align:
"left" });
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc
            .fontSize(12)
            .text("Telepon: ", { continued: true,
align: "left", width: 200 });
          doc.text(`${result.telepon}`, { align:
"left" });
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc.fontSize(12).text("Tanggal
Pemesanan: ", {
            continued: true,
            align: "left",
            width: 200,
          });
          doc.text(
            new
Date(result.tanggal_berkunjung).toLocaleTimeString
("id-ID", {
              hour: "2-digit",
              minute: "2-digit",
            }),
            { align: "left" }
          );
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc.fontSize(12).text("Jam Berkunjung:
", {
            continued: true,
            align: "left",
            width: 200,
```

```
            });
            doc.text(`${time}`, { align: "left" });
            doc.moveDown(); // Pindah ke baris
berikutnya

            doc.fontSize(12).text("Jumlah
Pengunjung: ", {
              continued: true,
              align: "left",
              width: 200,
            });
            doc.text(`${result.jumlah_orang}`, {
align: "left" });
            doc.moveDown(); // Pindah ke baris
berikutnya

            doc.fontSize(12).text("Kategori Dituju:
", {
              continued: true,
              align: "left",
              width: 200,
            });
            doc.text(`${result.category_dituju}`, {
align: "left" });
            doc.moveDown(); // Pindah ke baris
berikutnya

            doc.fontSize(12).text("Objek Dituju: ",
{
              continued: true,
              align: "left",
              width: 200,
            });
            doc.text(`${result.object_dituju}`, {
align: "left" });
            doc.moveDown(); // Pindah ke baris
berikutnya

            doc.fontSize(12).text("Total Pembayaran:
", {
              continued: true,
              align: "left",
```

```
            width: 200,
          });
          doc.text(
            `Rp${result.harga
              ?.toString()
              ?.replace(/\B(?=(\d{3})+(?!\d))/g,
".")}`,
            { align: "left" }
          );
          doc.moveDown(); // Pindah ke baris
berikutnya

          doc.fontSize(12).text("Status
Pembayaran: ", {
            continued: true,
            align: "left",
            width: 200,
          });
          doc.text(`${result.status}`, { align:
"left" });
          doc.moveDown(); // Pindah ke baris
berikutnya

          // Simpan file PDF ke dalam buffer
          const buffers = [];
          doc.on("data", (buffer) =>
buffers.push(buffer));
          doc.on("end", () => {
            const pdfBuffer =
Buffer.concat(buffers);

            // Tulis file PDF ke dalam sistem file
            fs.writeFile(`${result.uid}.pdf`,
pdfBuffer, (err) => {
              if (err) {
                console.error(err);
                return res.status(500).send("Gagal
membuat file PDF");
              }

              // Set header dan kirim file PDF
sebagai respons
```

	`` res.setHeader("Content-Disposition", 'attachment; filename="detail_pemesanan.pdf"'); res.setHeader("Content-Type", "application/pdf"); res.send(pdfBuffer); }); }); // Akhiri dan tutup objek PDFDocument doc.end(); } else { res.status(404).json({ message: "data not found", status: 404, data: [], }); } }); } catch (error) { res.status(400).json({ message: error.message, status: 400, data: [], }); } }; ```
Create Booking	``` controller.create = async (req, res) => { try { await model.booking .create({ uid: uid(), nama: req.body.nama, email: req.body.email, telepon: req.body.telepon, tanggal_berkunjung: req.body.tanggal_berkunjung, jam_berkunjung: req.body.jam_berkunjung, jumlah_orang: req.body.jumlah_orang, ```

```
        category_dituju: req.body.category_dituju,
        object_dituju: req.body.object_dituju,
        createdAt: new Date().getTime(),
        updatedAt: new Date().getTime(),
        bukti_pembayaran: req.file.filename,
        status: "pending",
        user_uid: req.body.user_uid,
        harga: req.body.harga,
      })
      .then((result) => {
        res.status(200).json({
          message: "berhasil menambahkan booking",
          status: 200,
          data: result,
        });
      });
  } catch (error) {
    res.status(404).json({
      message: error.message,
      status: 404,
      data: [],
    });
  }
};
```

Update Booking

```
controller.update = async (req, res) => {
  const data = {
    nama: req.body.nama,
    email: req.body.email,
    telepon: req.body.telepon,
    tanggal_berkunjung:
req.body.tanggal_berkunjung,
    jam_berkunjung: req.body.jam_berkunjung,
    jumlah_orang: req.body.jumlah_orang,
    category_dituju: req.body.category_dituju,
    object_dituju: req.body.object_dituju,
    updatedAt: new Date().getTime(),
    status: req.body.status,
  };
  try {
    await model.booking
      .findOne({
```

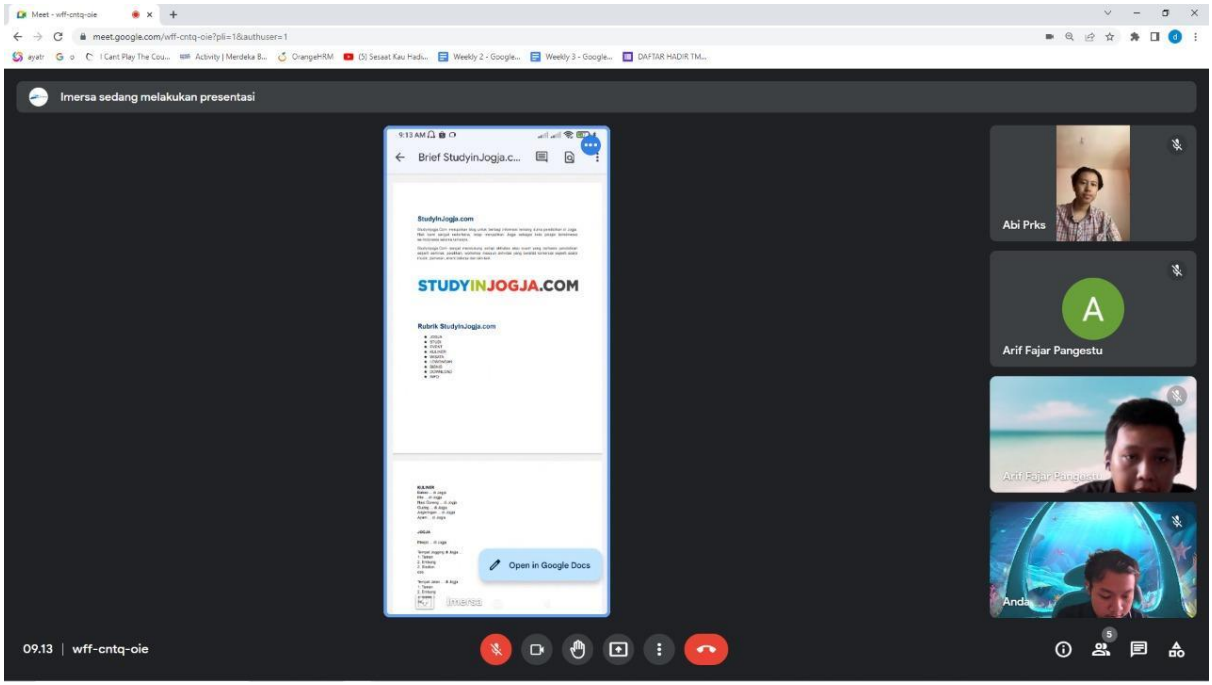
```
        where: {
          uid: req.params.id,
        },
      })
      .then((result) => {
        if (result) {
          model.booking
            .update(data, {
              where: {
                uid: req.params.id,
              },
            })
            .then((result) => {
              res.status(200).json({
                message: "berhasil update data",
                status: 200,
                data: result,
              });
            });
        } else {
          res.status(404).json({
            message: "data not found",
            status: 404,
            data: [],
          });
        }
      });
  } catch (error) {
    res.status(400).json({
      message: error.message,
      status: 400,
      data: [],
    });
  }
};
```

Delete Account

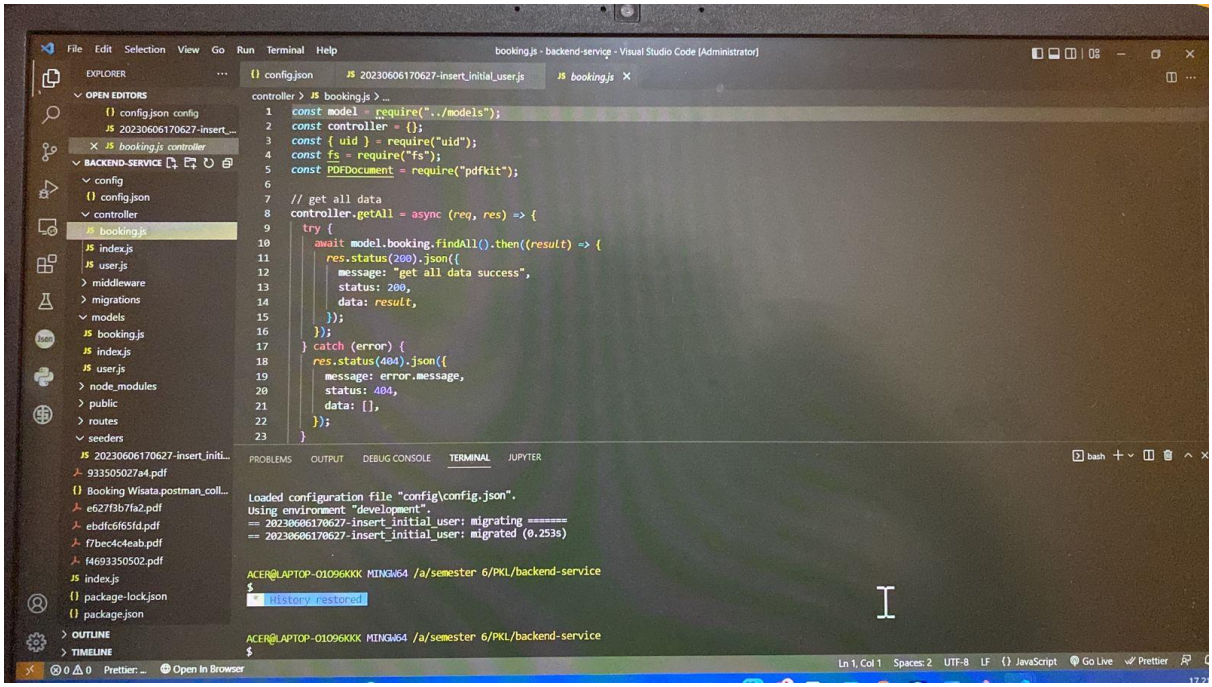
```
controller.delete = async (req, res) => {
  try {
    await model.booking
      .findOne({
        where: {
          uid: req.params.id,
```

```
        },
      })
      .then((result) => {
        if (result) {
          model.booking
            .destroy({
              where: {
                uid: req.params.id,
              },
            })
            .then((result) => {
              res.status(200).json({
                message: "berhasil hapus data",
                status: 200,
                data: result,
              });
            });
        } else {
          res.status(404).json({
            message: "data not found",
            status: 404,
            data: [],
          });
        }
      });
  } catch (error) {
    res.status(400).json({
      message: error.message,
      status: 400,
      data: [],
    });
  }
};
```

Lampiran 3: Dokumentasi Briefing Pengerjaan



Lampiran 4: Dokumentasi Pengerjaan



Lampiran 5: Surat Keterangan Selesai PKL

PT IMERSA SOLUSI TEKNOLOGI
Jl. Puntodewo 2 Baron, Nganjuk, Jawa Timur
Telp. 085755896233 E-mail : mail@imersa.co.id

Nomor : 017/PR.IMERSA/INTERNSHIP/I/2023
Lampiran : -
Hal : Surat Keterangan Selesai PKL

SURAT KETERANGAN

Yang bertanda tangan dibawah ini:

Nama : Miftahur Roziqin, S.Kom

Jabatan : Direktur PT Imersa Solusi Teknologi

Dengan ini menerangkan bahwa mahasiswa dengan identitas dibawah ini:

Nama : Muhammad Abi Prakosa
NPM : 20081010232

Nama : Arif Fajar Pangestu
NPM : 20081010236

Nama : Eragradiansyah Hardianto
NPM : 20081010218

Telah menyelesaikan kegiatan Praktik Kerja Lapangan (PKL) di PT Imersa Solusi Teknologi. Praktek Kerja Lapangan dilakukan selama 1 bulan terhitung mulai tanggan 9 Januari – 9 Februari 2023. Selama melakukan kegiatan PKL, mahasiswa tersebut telah bekerja dengan baik.

Nganjuk, 9 Februari 2023
Direktur
PT Imersa Solusi Teknologi

Miftahur Roziqin, S.Kom