

Dialog Presentasi - TIM THE FOOL

Fikri Production - Sistem Pemesanan Online

Tim: THE FOOL

Anggota:

1. Muhamad Dafin - Backend & Database
2. Ahmad Amru - Frontend & Routes
3. Fikri Ahmad - Deployment & Dokumentasi

Durasi: 15 menit (12 min presentasi + 3 min Q&A)

SLIDE 1: OPENING (30 detik)

[Muhamad Dafin berdiri]

Dafin:

"Selamat pagi/siang Bapak/Ibu. Perkenalkan, kami dari Tim THE FOOL akan mempresentasikan project Fikri Production - Sistem Manajemen Pemesanan Online untuk Unit Produksi Digital Printing SMKIT Ihsanul Fikri.

Tim kami terdiri dari:

- Muhamad Dafin, saya bertanggung jawab untuk backend dan database development
- Ahmad Amru, menangani frontend dan routing system
- Fikri Ahmad, fokus pada deployment dan dokumentasi

Project ini adalah full-stack web application yang bertujuan untuk digitalisasi proses pemesanan printing di unit produksi sekolah."

[Next slide]

SLIDE 2: LATAR BELAKANG MASALAH (2 menit)

Dafin:

"Baik, saya akan jelaskan latar belakang dan problem yang kami solve dengan aplikasi ini.

Masalah yang ada saat ini:

Unit Produksi masih menggunakan sistem manual. Customer harus datang langsung untuk pesan, tidak ada sistem tracking pesanan, sering terjadi antrian panjang terutama saat deadline tugas, dan file design yang dikumpulkan via flashdisk sering hilang atau rusak.

Ini menciptakan inefficiency baik untuk customer maupun admin. Customer harus menyesuaikan waktu dengan jam operasional, dan admin kesulitan mengelola data pesanan yang masih dicatat manual di buku.

Solusi yang kami tawarkan:

Web application yang bisa diakses 24/7. Customer bisa order kapan saja, upload file design langsung via website, dan tracking status pesanan secara real-time.

Untuk admin, semua data tersimpan di database yang terorganisir, ada dashboard untuk manage pesanan, dan bisa export data ke Excel atau PDF untuk reporting.

Sistem ini juga include auto-backup untuk files, jadi tidak ada lagi masalah file hilang.

Target pengguna aplikasi ini adalah siswa dan guru SMKIT Ihsanul Fikri untuk kebutuhan printing akademis, masyarakat umum yang butuh jasa printing, dan staff Unit Produksi sebagai admin untuk kelola pesanan."

[Next slide]

SLIDE 3: ARSITEKTUR SISTEM (1.5 menit)

Dafin:

"Sekarang saya jelaskan arsitektur teknis dari aplikasi kami.

Kami menggunakan three-layer architecture yang cukup standard untuk web application:

Layer pertama adalah Frontend - Ini adalah interface yang dilihat user. Kami pakai HTML5 dan CSS3 untuk structure dan styling, JavaScript untuk interactivity, dan Jinja2 sebagai templating engine untuk render dynamic content dari backend.

Layer kedua adalah Flask Application - Ini adalah core logic. Kami split code jadi tiga file utama:

- `routes.py` untuk handle HTTP requests dan responses
- `models.py` untuk define database schema
- `app.py` untuk configuration dan initialization

Kenapa split? Supaya code lebih organized dan kalo ada update, kita bisa work di file yang berbeda without conflict.

Layer ketiga adalah Database - Kita pakai SQLite untuk development karena lightweight dan no setup needed. Tapi setelah deploy ke Railway, otomatis migrate ke PostgreSQL yang lebih robust untuk production.

Dan terakhir File Storage - Semua file yang diupload customer disimpan di folder `uploads/` dengan proper naming convention untuk avoid conflicts.

Data flow-nya simple: User submit form → Flask process request → Save to database → Store file if any → Return response.

Arsitektur ini modular, scalable, dan follow best practices untuk web development."

[Next slide]

SLIDE 4: FITUR APLIKASI (2 menit)

[Ahmad Amru mengambil alih]

Amru:

"Terima kasih Dafin. Sekarang saya akan explain fitur-fitur yang ada di aplikasi.

Fitur untuk User:

Pertama, Landing Page - Halaman utama yang menampilkan katalog produk. Ada 6 produk: Document Printing yang sudah available, dan 5 merchandise products yang coming soon seperti custom t-shirt, mug, sticker, dll. Design-nya modern dengan organic shapes dan smooth animations.

Kedua, Form Pemesanan - User bisa input data mereka: nama, kontak, pilih jenis print, ukuran kertas, jumlah, dan upload file design. Form ini ada validation baik di frontend maupun backend untuk ensure data quality.

Ketiga, Cek Status - Setelah order, user dapat nomor pesanan. Mereka bisa cek status kapan saja dengan input nomor tersebut. Status akan update real-time: pending, diproses, atau selesai.

Fitur untuk Admin:

Pertama, Login System - Admin authentication menggunakan session-based. Secure dan simple.

Kedua, Dashboard - Central hub untuk manage semua pesanan. Admin bisa lihat list orders, ada statistics di atas showing total pesanan, pending, in-progress, dan completed. Bisa filter by status dan search by nama atau kontak.

Ketiga, Management - Admin bisa update status pesanan dengan dropdown, dan delete pesanan jika needed. Semua perubahan langsung tersimpan di database.

Keempat, Export Features - Ini penting untuk reporting. Admin bisa export data ke Excel dengan styling yang professional, atau ke PDF untuk print reports. File otomatis ter-download dengan timestamp di filename.

Semua fitur ini dirancang untuk user-friendly baik untuk customer maupun admin."

[Next slide]

SLIDE 5: TECHNOLOGY STACK (1.5 menit)

Amru:

"Untuk technology stack yang kami gunakan:

Backend:

- Python 3.14 sebagai programming language
- Flask 3.0.0 sebagai web framework - kami pilih Flask karena lightweight dan flexible
- SQLAlchemy untuk ORM - ini abstraction layer yang memudahkan database operations
- SQLite untuk development, PostgreSQL untuk production di Railway

Frontend:

- HTML5 dan CSS3 untuk markup dan styling
- Vanilla JavaScript - kita tidak pakai framework seperti React karena aplikasi ini relatif simple dan performance jadi lebih cepat
- Jinja2 templating - integrated dengan Flask untuk dynamic content rendering
- Design-nya AI-assisted menggunakan Claude untuk speed up development, tapi semua functionality kami code manual

Libraries:

- openpyxl untuk generate Excel files dengan styling
- reportlab untuk PDF generation
- Werkzeug untuk security utilities seperti secure_filename
- Gunicorn sebagai WSGI server untuk production

Deployment:

- Railway.app sebagai hosting platform - kenapa Railway? Karena:
 - Auto-deploy langsung dari GitHub push
 - Built-in PostgreSQL database
 - HTTPS certificate otomatis
 - Zero configuration needed
 - Free tier cukup untuk project ini

Stack ini reliable, proven, dan suitable untuk production use."

[Next slide]

SLIDE 6: PEMBAGIAN TUGAS (2 menit)

[Fikri Ahmad mengambil alih]

Fikri:

"Terima kasih Amru. Sekarang saya akan explain pembagian tugas dan workflow team kami.

Muhammad Dafin - Backend & Database:

Dafin bertanggung jawab untuk:

- Design database schema di `models.py` - mendefinisikan table structure, columns, data types, dan constraints
- Implement CRUD operations - Create, Read, Update, Delete untuk pesanan
- Setup SQLAlchemy ORM dan connection handling
- Develop backend API endpoints logic yang akan dipanggil dari routes

Deliverable dari Dafin: `models.py` yang complete dengan Pesanan model, helper functions, dan database initialization script.

Ahmad Amru - Frontend & Routes:

Amru handle:

- Implementation semua routes di `routes.py` - total ada 9 endpoints: 3 public routes untuk user, 6 admin routes
- Create HTML templates - landing page, order form, status check, admin dashboard
- Styling dengan CSS - responsive design yang work di desktop dan mobile
- Form validation both client-side dan server-side
- Export features implementation - Excel dan PDF generation

Deliverable: `routes.py` complete, folder templates/ dengan semua HTML files, dan static assets.

Fikri Ahmad - Deployment & Documentation:

Saya responsible untuk:

- Setup Railway deployment - connect GitHub repository, configure build settings, environment variables
- Database migration dari SQLite ke PostgreSQL di production
- Create comprehensive documentation - README, deployment guide, troubleshooting
- Testing application end-to-end
- Handle any deployment issues atau bugs

Deliverable: Deployed application yang accessible via public URL, complete documentation, dan test results.

Timeline Development:

Week 1: Dafin fokus ke database, Amru mulai public routes, saya setup repository

Week 2: Dafin finalize CRUD, Amru develop admin features, saya prepare Railway

Week 3: Integration testing, bug fixes, deployment, documentation

Total development time: 3 minggu dengan regular coordination via WhatsApp dan weekly meetings."

[Next slide]

❖ SLIDE 7: GITHUB WORKFLOW (2 menit)

Fikri:

"Untuk collaboration, kami menggunakan Git dan GitHub dengan proper workflow.

Repository Structure:

Repository kami ada di `github.com/thepool/fikri-production` dengan structure yang organized:

- `app.py`, `models.py`, `routes.py` di root

- Folder `templates/` untuk HTML files
- Folder `static/` untuk CSS, images
- `requirements.txt` untuk dependencies
- `README.md` untuk documentation

Git Workflow yang kami follow:

Step 1: Setiap anggota clone repository dan create branch sendiri. Contoh:

- Dafin: `feature/database-models`
- Amru: `feature/user-routes`
- Saya: `feature/deployment-config`

Step 2: Development di branch masing-masing. Commit frequently dengan descriptive messages. Contoh:

"Add Pesanan model with validation" atau "Implement order form with file upload".

Step 3: Push branch ke remote repository.

Step 4: Create Pull Request ke main branch. Di PR description, kami explain what changes made dan why.

Step 5: Code review by team members. Minimal 1 approve before merge. Kita check:

- Code quality dan readability
- Potential bugs atau errors
- Consistency dengan existing code
- Documentation comments

Step 6: Setelah approved, merge to main. Railway automatically detect push dan trigger deployment.

Dalam 2-3 menit, changes sudah live di production.

Requirements Penilaian yang kami meet:

- ✓ **Minimal 20 commits** - Kami punya 25+ commits total, distributed evenly across team members
- ✓ **Pull request dan merge** - Setiap feature development through PR process. We have documented code reviews dan merge history.
- ✓ **Kontribusi setiap anggota** - GitHub contribution graph show balanced activity from all three members. No single-person dominance.
Git workflow ini ensure code quality, prevent conflicts, dan maintain clean history. Plus, auto-deploy from GitHub to Railway make deployment seamless."

[Next slide]

❖ SLIDE 8: DEMO & DEPLOYMENT (2 menit)

Fikri:

"Sekarang tentang deployment dan demo.

Railway.app Deployment:

Kenapa kami pilih Railway instead of setup server manual?

Pertama, simplicity - No need configure Nginx, Gunicorn, SSL certificates manually. Railway handle semua itu automatically.

Kedua, GitHub integration - Setiap kali kami push to main branch, Railway auto-detect changes dan deploy. No manual steps needed. Build time sekitar 2-3 menit.

Ketiga, database - Railway provide PostgreSQL database built-in. Connection string automatically injected as environment variable. Migration dari SQLite ke PostgreSQL seamless karena kami pakai SQLAlchemy ORM.

Keempat, HTTPS - Railway automatically generate SSL certificate and serve application over HTTPS. Secure by default.

Kelima, environment variables - Kita bisa configure sensitive data seperti secret keys via Railway dashboard, tidak hardcode di code.

Zero-downtime deployment - Railway use rolling deployment strategy. Old version keep running until new version ready, then switch. User tidak experience downtime.

Live Demo:

[Buka browser dan navigate ke Railway URL]

Ini adalah aplikasi kami yang sudah deployed. URL-nya adalah [railway-url].

[Demo Landing Page] Ini landing page dengan design modern. Ada 6 product cards, navigation smooth, responsive di mobile.

[Click Order Now] Form pemesanan dengan all fields required. Saya isi sample data... [Fill form]... dan upload file PDF.

[Submit] Setelah submit, dapat konfirmasi dan nomor pesanan.

[Navigate to Cek Status] Input nomor pesanan... dan bisa lihat status real-time. Status saat ini 'Pending'.

[Navigate to Admin Login] Login dengan credentials... masuk ke dashboard.

[Show Dashboard] Dashboard showing statistics di atas, table pesanan di bawah. Bisa filter by status, search by name.

[Update status dari Pending to Proses]

Status terupdate, dan kalau user cek status lagi, akan terlihat perubahan immediately.

[Click Export Excel] Excel file ter-download dengan data terformat professional.

Ini adalah end-to-end workflow from order to completion, semua running smooth di production environment di Railway."

[Next slide]

SLIDE 9: CLOSING (30 detik)

[Muhamad Dafin returns untuk closing]

Dafin:

"Baik, sebagai penutup.

Project Fikri Production ini adalah solusi digitalisasi untuk Unit Produksi sekolah. Dengan aplikasi ini, proses pemesanan jadi lebih efficient, transparent, and convenient untuk semua pihak.

Key achievements:

- Full-stack web application dengan clean architecture
- Modular code structure untuk better maintainability
- Deployed di Railway dengan auto-deployment dari GitHub
- Complete documentation dan testing
- Git workflow dengan proper PR review process

Project deliverables:

- Working application: [railway-url]
- GitHub repository: github.com/theppool/fikri-production
- Documentation: README, deployment guide
- 25+ commits dengan balanced contributions
- Multiple merged pull requests dengan code reviews

Terima kasih atas perhatiannya. Kami sekarang terbuka untuk pertanyaan."

[Q&A Session]

ANTICIPATED Q&A

Q1: Kenapa pilih Railway instead of deploy manual di server?

Fikri:

"Good question. Ada beberapa alasan:

Pertama, time efficiency - Setup Nginx, Gunicorn, SSL certificates manually butuh banyak time dan troubleshooting. Railway eliminate semua complexity itu.

Kedua, maintenance - Dengan manual server, kita harus handle server updates, security patches, monitoring sendiri. Railway handle infrastructure management, kita fokus ke application development.

Ketiga, scalability - Kalau traffic naik, di Railway kita tinggal upgrade plan. Di manual server, harus provision new servers dan setup load balancing sendiri.

Keempat, CI/CD - Auto-deploy dari GitHub built-in di Railway. Kalau manual, harus setup GitHub Actions atau Jenkins sendiri.

Kelima, PostgreSQL - Railway provide managed PostgreSQL. Di manual server, harus install, configure, backup PostgreSQL sendiri.

For production applications especially untuk project scope seperti ini, managed platform seperti Railway adalah smart choice. Trade-off adalah cost (Railway paid setelah free tier habis), tapi benefit dalam terms of time saved dan reduced operational complexity worth it."

Q2: Bagaimana handle file upload security?

Dafin:

"Security untuk file upload kami implement di several layers:

Layer 1: Extension validation - Kami hanya allow specific extensions: PDF, PNG, JPG, JPEG, DOCX. Extensions lain akan direject.

Layer 2: Filename sanitization - Kami pakai `secure_filename()` dari Werkzeug library. Function ini remove special characters, path separators, dan anything yang potentially dangerous. Contoh: `../../../../etc/passwd` akan jadi `etc_passwd`.

Layer 3: File size limit - Config `MAX_CONTENT_LENGTH` set to 16MB. Files lebih besar akan automatically rejected oleh Flask.

Layer 4: Storage segregation - Uploaded files disimpan di `uploads/` directory yang separate dari application code. Files tidak directly accessible via URL, harus through Flask route yang bisa add access control.

Layer 5: Unique naming - Kami tambah timestamp prefix ke filename:

`YYYYMMDD_HHMMSS_originalname.ext`. Ini prevent filename collision dan make files trackable.

Future enhancements yang bisa ditambah:

- Virus scanning untuk uploaded files
- File type validation beyond extension (check magic bytes)
- Content scanning untuk potentially malicious content
- Rate limiting untuk prevent upload abuse

Current implementation sufficient untuk use case kami, tapi always room untuk improvement dalam security."

Q3: Kalau ada conflict saat merge di GitHub, gimana handle-nya?

Fikri:

"Git conflicts memang inevitable dalam team development. Ini adalah workflow kami:

Prevention:

- Coordination sebelum work - kita communicate di WhatsApp file mana yang akan di-edit
- Modular architecture - `models.py`, `routes.py`, `app.py` separate, reduce chance of simultaneous edits
- Frequent pulls - Setiap mulai work, pull latest changes dulu dari main branch
- Small commits - Commit frequently dengan atomic changes, easier to merge

When conflict happens:

Step 1: Git akan mark conflict di file:

<<<<< HEAD

current code

=====

incoming changes

>>>>> feature-branch

Step 2: Manual resolution - Kita discuss di WhatsApp atau call, decide mana yang keep, mana yang modify. Usually kami collaborate real-time untuk resolve.

Step 3: Test after resolution - Run application, ensure nothing broken.

Step 4: Commit resolution dan push.

Experience: Kami actually encounter conflict sekali di `routes.py` ketika Amru dan saya simultaneously add different routes. Resolution straightforward karena routes independent, kita keep both changes.

Key adalah communication. Good team coordination minimize conflicts significantly."

Q4: Berapa biaya running aplikasi di Railway per bulan?

Fikri:

"Untuk Railway pricing:

Free Tier:

- \$5 credit per month (sufficient untuk ~500 hours uptime)
- 1GB RAM
- PostgreSQL database included
- HTTPS included
- Perfect untuk project showcase atau low-traffic apps

Untuk aplikasi kami dengan **current traffic** (estimated ~50 users per day, ~200 requests per day), free tier more than enough. Aplikasi kami probably use ~\$2-3 per month dari free credit.

Paid Plans (jika scaling needed):

- Hobby: \$5/month untuk remove sleep mode
- Pro: \$20/month dengan more resources dan priority support

Cost comparison:

- Railway (\$0-5/month): Managed platform, no maintenance
- VPS (Digital Ocean, ~\$5-10/month): Harus maintenance sendiri
- Heroku (\$7/month minimum): Similar to Railway tapi lebih mahal

Break-even analysis: Kalau traffic increase significantly (>1000 users/day), mungkin make sense migrate to VPS untuk cost efficiency. Tapi untuk current scale, Railway optimal karena zero maintenance overhead.

Important: Railway free tier sufficient for portfolio/demo purposes. Jika aplikasi actually adopted by Unit Produksi dengan heavy daily usage, sekolah could sponsor hosting cost which is minimal (\$5-20/month)."

Dafin:

"Ada pertanyaan lain?"

[Continue Q&A as needed...]

TIPS PRESENTASI

Persiapan:

- Test Railway deployment 1 jam sebelum presentasi (pastikan running)
- Bookmark URL di browser
- Test demo flow (order → status → admin) sebelumnya
- Charge laptop full
- Bawa adapter HDMI untuk proyektor

During Presentasi:

- Speak clearly dan tidak terlalu cepat
- Make eye contact dengan audience
- Jangan membaca slide verbatim, explain dengan bahasa sendiri
- Kalau live demo error, calm down dan explain "in real production this works" + show screenshots

Q&A Tips:

- Kalau tidak tahu jawaban: "That's a good question. Honestly kami belum explore that aspect. Would be good future improvement."
 - Kalau pertanyaan teknis heavy: "Let me explain the architecture..." then redirect to relevant slide
 - Encourage questions dengan "Ada yang ingin ditanyakan?"
-

END OF PRESENTATION

TIM THE FOOL

Muhamad Dafin • Ahmad Amru • Fikri Ahmad

SMKIT Ihsanul Fikri Mungkid