Go + Flutter Course

Rapid Cross-Platform Development

Timur Harin

Lecture 00: Introduction

Who am I? Why I can teach you?

Timur Harin

- Head of Software Development Department in Center of Autonomous Technologies in Innopolis University
- My interest in teaching how to use modern technologies to solve real-world problems
- Twice taught course about flutter development in fall 2024 and summer 2024

My research interests

- Software engineering - what tools to use to build software and their best practices
- Software architecture - how to design software, programing languages, frameworks, libraries, etc.
- Software construction - what are existing components and how to use them in software development

Why Are We Here?

This course bridges the gap between academic programming and industry reality

- You are in the *beginning of your career*, hard to choose what to learn and what to focus on
- Learn **production-ready** tools used in big tech companies
- Master **full-stack development** with modern technologies
- Practice **real software engineering workflows** - how people actually work in software development
- Build **portfolio-worthy projects** - you can show them to your future employers, because all in github

The AI Era: Why Rapid Development Matters

We're Living in the AI Revolution

- **Hypothesis-driven development** is crucial - test ideas quickly
- **Rapid prototyping** separates successful projects from failures
- **Speed to market** determines competitive advantage
- **Iterative learning** from real user feedback

Modern Development Demands

- **Build fast, learn fast, iterate fast**
- **Cross-platform reach** maximizes impact
- **Efficient backends** handle AI workloads and user scale
- **Quick validation** of product-market fit

Why This Elective?

The Modern Development Landscape

- **Cross-platform** is the future (Mobile, Web, Desktop, Embedded)
- **Microservices** architecture dominates backend development, because of the complexity of the software and the need for scalability
- **DevOps** and **CI/CD** are essential skills
- **Code review** and **collaboration** are daily practices

Skills That Matter

- Go: Performance + Simplicity
- Flutter: One codebase, multiple platforms
- Real workflow: Git, MR, peer review

Why Go + Flutter?

Go (Backend)

- **Fast compilation** and **excellent performance**
- **Built-in concurrency** with goroutines
- **Simple syntax** but **powerful features**
- **Excellent tooling** and **strong standard library**

Flutter (Frontend)

- **Single codebase** for Mobile, Web, Desktop, Embedded
- **Native performance** with beautiful UIs
- **Hot reload** for rapid development
- **Growing ecosystem** and **Large community**

Big Tech Companies Using These Tools

Go Users

- **Google**: Kubernetes, Docker, Go itself
- **Uber**: Backend services, microservices
- **Netflix**: Performance-critical services
- **Cloudflare**: Edge computing
- **Ozon, Yandex, Wildberries**: Backend services, microservices
- ...

Flutter Users

- **Google**: Google Ads, Google Pay
- **Alibaba**: Xianyu app (50M+ users)
- **BMW**: My BMW app
- **eBay**: eBay Motors
- **Tencent**: Multiple apps
- **Xiaomi**: Multiple apps
- **Yandex Pro**: Yandex.Taxi app
- ...

Course Structure: 8 Intensive Blocks

Each Block = 1 Lecture (1.5h) + 1 Lab (1.5h)

1. **Foundations** - Go basics + Flutter widgets
2. **Concurrency & Streams** - Goroutines + async programming
3. **Data & APIs** - HTTP servers + REST clients
4. **Database & Persistence** - PostgreSQL + local storage
5. **Advanced Patterns & Testing** - Clean architecture + testing
6. **Authentication & Security** - JWT + secure storage
7. **WebSockets & gRPC** - Real-time communication
8. **Docker & Production** - Containerization + deployment

Schedule Flexibility & Time Management

Adaptive Schedule

- **Larger amount of dates** allocated for each block
- **Flexible scheduling** when instructor cannot deliver on planned date
- **All sessions will be moved** - nothing gets cancelled
- **Advanced notice** provided for any changes

Communication

- **Telegram notifications** for schedule updates

Life happens - we adapt and keep learning!

What You'll Build

Final Project

A **complete full-stack application** with:

- Go backend with microservices
- Flutter frontend (mobile + web)
- Database integration
- Authentication & authorization
- Real-time features
- Containerized deployment

Grading & Assessment (part 1)

Lab Assignments (42%)

- **7 labs** with progressive complexity
- **Working software** that meets requirements
- **Automated testing** and **code quality** checks
- **Documentation** and **clean code**

Peer Review Participation (14%)

- **Quality of feedback** provided to classmates
- **Timeliness** of reviews (within 48 hours)
- **Constructive criticism** and **helpful suggestions**
- **Minimum 2 reviews per lab** required

Grading & Assessment (part 2)

Final Project (30%)

- **Comprehensive application** using all course concepts
- **Documentation** and **deployment**
- **What you found new** and **what you learned**
- **What you would do differently**

Attendance & Participation (14%)

- **7 lectures and 7 labs attendance** (mandatory, I want to see you)

Grading & Assessment (part 3)

Bonus Points (up to +10%)

- **In-class collaboration** - helping classmates debug issues
- **Active participation** in discussions - asking thoughtful questions
- **GitHub contributions** for course improvement - fixing typos, suggestions
- **Community engagement** - sharing resources, writing blog posts

Grading & Assessment (part 4)

Excellence Track: Automatic Grade A

- **Alternative path**: Complete challenging individual project + technical interview. According future technical requirements document

Technical Requirements:

- **Full-stack application** with Go backend + Flutter frontend
- **Microservices architecture** (minimum 3 services)
- **Advanced features**: WebSockets, gRPC, caching, monitoring
- **Production deployment** with Docker + CI/CD pipeline
- **Comprehensive testing** + technical documentation
- **Interview:** 30-minute technical discussion + code walkthrough
- **Deadline:** End of Block 6

Late Submission Policy

Lab Deadlines

- **Each lab has a defined deadline** announced with the assignment
- **Peer reviews** have separate deadlines (typically 48h after lab deadline)
- **No extensions** without proper documentation

Grading Timeline

- **You submit on time → I grade on time → You pass**
- **You submit late → Automatic zero** (no exceptions)
- **I grade late → You get full points** (instructor accountability)

Exception Process

- **Documentation required** for emergencies (medical, family, etc.)
- **Dean's office confirmation** needed for official excuses
- **Resubmission opportunity** only with proper documentation

AI Usage Policy & Learning Philosophy

AI as a Learning Tool

- **Encouraged**: Use AI to **understand concepts** and **explore solutions**
- **Required**: **Go into details** - understand what the AI suggests
- **Critical**: **Don't auto-accept** - question, verify, and improve
- **Essential**: **Write tests** for AI-generated code

Best Practices

- **Use AI to explain** complex concepts you're struggling with
- **Ask AI to suggest** alternative approaches to problems
- **Have AI review** your code for potential improvements
- **Always test and validate** AI suggestions thoroughly
- **Document your understanding** - explain the code in your own words

What We Don't Want

- Copy-paste without understanding
- Blind trust in AI outputs
- Skipping the learning process

My tools

- **Cursor+MCP**: Code
- **ChatGPT**: AI for big problem solving
- **DeepSeek or Qwen**: AI for small problems solving
- **Perplexity**: for searching and analyzing

Feedback & Continuous Improvement

Your Voice Matters

- **Open to feedback** throughout the course
- **Multiple feedback forms** will be shared in Telegram chat
- **Quick surveys** after each major block
- **Course improvement** based on your input

Feedback Policy

- **Please fill out forms** when posted - your input shapes the course
- **Initially optional** but highly encouraged
- **If participation is low** → forms become **mandatory before solution submission**
- **Anonymous options** available for sensitive feedback

What We'll Ask About

- Lecture pace and clarity
- Lab difficulty and relevance
- Tool effectiveness
- Suggested improvements

Grade Thresholds

Letter Grade Breakdown

- **A (90-100%)**: Exceptional work, exceeds expectations
- **B (70-89%)**: Good work, meets all requirements
- **C (60-69%)**: Satisfactory work, meets most requirements
- **F (0-59%)**: Failing, major requirements not met

Component Breakdown Reminder

- **Labs**: 42% (7 labs × 6% each)
- **Peer Reviews**: 14% (2 reviews per lab)
- **Final Project**: 30%
- **Attendance**: 14%
- **Bonus**: up to +10%

Peer Review Management System

Assignment Strategy

- **Round-robin rotation**: Each student reviews different classmates each week
- **Random pairing**: Automated assignment via spreadsheet
- **No self-reviews**: Cannot review your own team members' code

Review Requirements

- **Minimum 2 reviews** per lab submission
- **48-hour deadline** after submission
- **Structured feedback** using provided template
- **Check tests and code quality** before reviewing

Peer Review Tools & Implementation

Review Template (Required Format)

```
## Code Review for LabXX - [Student Name]

### ✅ What Works Well
- Feature X implementation is clean
- Good error handling in function Y

### 🔍 Issues Found
- Bug in line 45: null pointer exception
- Missing unit tests for core functionality

### 💡 Suggestions
- Consider using dependency injection
- Add input validation

### 🧪 Testing
- [ ] Code compiles successfully
- [ ] All requirements met
- [ ] No security issues found
```

Tracking System

- **Source**: Will be announced later

Lab Workflow: Like Real Software Engineering

1. Fork & Clone

```
git clone https://github.com/your-username/sum25-go-flutter-course.git
```

2. Create Branch

```
git checkout -b lab01-surname-name
```

3. Complete Lab

- Write code in `labs/labXX/` directory
- Ensure tests pass: `make test`
- Check code quality: `make lint`

Merge Request Process 🔍

4. Submit for Review

```
git push origin lab01-surname-name
# Create Merge Request on GitHub
```

5. Automated Checks

- ✅ **CI Pipeline**: Tests, linting, builds
- ✅ **Code Quality**: Style, documentation

6. Peer Review

- **Required**: At least one peer review
- **Encourage**: Constructive feedback
- **Learn**: From others' solutions

7. Instructor Review & Merge

- Will be announced later

Why This Workflow?

Real Industry Practices

- **Code review** catches bugs and improves quality
- **CI/CD** ensures reliability and consistency
- **Collaboration** develops communication skills
- **Git workflow** is industry standard

Learning Benefits

- **See different approaches** to same problems
- **Learn from mistakes** before they reach production
- **Build professional habits** early
- **Network** with classmates

Getting Started Today

Prerequisites Check

- ✅ Git installed and configured
- ✅ Go 1.24.3+ installed
- ✅ Flutter 3.32.1+ installed
- ✅ Docker and Docker Compose
- ✅ PostgreSQL client

First Steps

1. **Fork** the course repository
2. **Clone** your fork locally
3. **Run** `make setup` to check dependencies and install them if needed
4. **Start** development environment: `make dev`

Course Resources 📖

Documentation

- Go Documentation
- Flutter Documentation
- Course Repository

Communication

- **Telegram chat** for technical and general questions
- **In-person labs** for hands-on help
- **Peer review** for collaborative learning

What Makes This Course Different?

Not Just Theory

- **Real tools** used in production
- **Professional workflows** from day one
- **Portfolio projects** you can show employers
- **Industry practices** that matter

Hands-On Learning

- **Build, don't just read** about concepts
- **Debug real problems** you'll face in industry
- **Collaborate** like professional developers
- **Deploy** to real environments

Questions?

Ready to build something amazing?

Next week: Lab 01 - Building your first Go program + Flutter widget

Come prepared with:

- Laptop with prerequisites installed
- GitHub account ready
- Questions about the setup

Let's Build the Future!

- **Remember**: Every expert was once a beginner
- Everyone can make mistakes, but the difference is that experts learn from them and never stop building
- **The difference**: They never stopped building

Welcome to your journey in modern software development!