

Building Oscar

A Step-by-Step Guide
Build your own AI email assistant using Claude

Presented by Aloa



Section 1: Install Prerequisites

What we're doing

Installing the tools needed to build Oscar.

The Prompt

Copy and paste this into Claude Code:

```
I need to set up my computer to build an AI agent. Please help me:  
  
1. Check if Node.js is installed (need v18+). If not, tell me how to install it.  
2. Install the Claude CLI globally: npm install -g @anthropic-ai/clause-code  
3. Run "claude login" and help me complete the authentication  
  
After each step, tell me what to do and wait for me to confirm before moving on.
```

Checkpoint

Run these commands in Terminal. Both should work without errors:

```
node --version  
# Should show v18.x.x or higher  
  
claude --version  
# Should show a version number
```

Troubleshooting

"node: command not found"

→ Download Node.js from <https://nodejs.org> (choose LTS version)

"claude: command not found"

→ Run: `npm install -g @anthropic-ai/clause-code`

Claude login fails

→ Make sure you have a Claude account at anthropic.com

Section 2: Create Project Structure

What we're doing

Creating the folder structure and configuration files for Oscar.

The Prompt

Copy and paste this into Claude Code:

Create a new project called "oscar" with this structure:

```
oscar/
  └── packages/
    ├── backend/
    │   └── src/
    ├── frontend/
    │   └── src/
  └── package.json (monorepo with workspaces)
  └── tsconfig.base.json
  └── .gitignore
  └── .env
```

For package.json, use:

```
{
  "name": "oscar",
  "private": true,
  "workspaces": [ "packages/*" ],
  "scripts": {
    "dev": "concurrently \"npm run dev:backend\" \"npm run dev:frontend\"",
    "dev:backend": "npm run dev --workspace=packages/backend",
    "dev:frontend": "npm run dev --workspace=packages/frontend"
  },
  "devDependencies": {
    "concurrently": "^8.2.2",
    "typescript": "^5.3.3"
  }
}
```

For tsconfig.base.json:

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "esModuleInterop": true,
    "strict": true,
    "skipLibCheck": true,
    "resolveJsonModule": true
  }
}
```

For .gitignore:

```
node_modules/
.env
```

```
*.log  
dist/  
.tokens.json  
  
For .env (template - I'll fill in values later):  
GOOGLE_CLIENT_ID=  
GOOGLE_CLIENT_SECRET=  
GOOGLE_REDIRECT_URI=http://localhost:3001/auth/google/callback  
ENCRYPTION_KEY=  
PORT=3001  
FRONTEND_URL=http://localhost:5173
```

Create all these files now.

Checkpoint

Run this in Terminal from your project folder:

```
ls -la  
# Should show: packages/, package.json, tsconfig.base.json, .gitignore, .env  
  
ls packages/  
# Should show: backend/, frontend/
```

Troubleshooting

Folders not created

→ Make sure you're in the right directory. Run `pwd` to see where you are.

Permission denied

→ You may need to create the oscar folder first: `mkdir oscar && cd oscar`

Section 3: Set Up Backend

What we're doing

Creating the backend server that will talk to Claude and Gmail.

The Prompt

Copy and paste this into Claude Code:

Set up the Oscar backend. Create these files:

1. packages/backend/package.json:

```
{  
  "name": "oscar-backend",  
  "type": "module",  
  "scripts": {  
    "dev": "tsx watch src/index.ts"  
  },  
  "dependencies": {  
    "@anthropic-ai/clause-code": "^1.0.3",  
    "cors": "^2.8.5",  
    "dotenv": "^16.4.5",  
    "express": "^4.21.0",  
    "googleapis": "^144.0.0",  
    "zod": "^3.23.8"  
  },  
  "devDependencies": {  
    "@types/cors": "^2.8.17",  
    "@types/express": "^4.17.21",  
    "@types/node": "^22.5.5",  
    "tsx": "^4.19.1"  
  }  
}
```

2. packages/backend/tsconfig.json:

```
{  
  "extends": "../../tsconfig.base.json",  
  "compilerOptions": {  
    "outDir": "./dist",  
    "rootDir": "./src"  
  },  
  "include": ["src/**/*"]  
}
```

3. packages/backend/src/index.ts - Express server with CORS, routes for /auth and /chat, health check

4. packages/backend/src/routes/auth.ts - OAuth routes:

- GET /google - redirects to Google OAuth
- GET /google/callback - handles OAuth callback
- GET /status - returns { authenticated, accounts }
- DELETE /disconnect/:email - disconnects one account
- DELETE /disconnect - disconnects all accounts

5. packages/backend/src/routes/chat.ts - Chat route:

- POST / - accepts { message, sessionId }
 - Uses Claude Agent SDK's query() function
 - Streams responses via Server-Sent Events
 - Uses createSdkMcpServer with tools (empty array for now)
 - System prompt: "You are Oscar, a helpful AI assistant that helps users manage their emails. Be cool."
 - Set maxThinkingTokens: 10000
6. packages/backend/src/services/tokenStore.ts - Encrypted token storage:
- Uses AES-256-GCM encryption
 - Functions: storeToken, getToken, getAllTokens, removeToken, getConnectedAccounts
 - Stores in .tokens.json file
7. packages/backend/src/services/gmail.ts - Gmail service:
- getAuthUrl() - generates Google OAuth URL
 - handleCallback(code) - exchanges code for tokens, stores them
 - getAllGmailClients() - returns Gmail API clients for all connected accounts
8. packages/backend/src/types/index.ts - TypeScript types

Create all these files with complete, working code.

Checkpoint

Run from the project root:

```
npm install
npm run dev:backend
```

You should see: Oscar backend running on port 3001

Test the health endpoint:

```
curl http://localhost:3001/health
# Should return: {"status": "ok"}
```

Troubleshooting

"Cannot find module"

→ Run `npm install` from the project root

Port 3001 in use

→ Run: `lsof -ti :3001`

xargs kill -9` then try again

TypeScript errors

→ Make sure tsconfig.json files are correct

Section 4: Create Gmail Tools

What we're doing

Adding tools that let Oscar read, search, and send emails.

The Prompt

Copy and paste this into Claude Code:

Create the Gmail tools for Oscar in packages/backend/src/agent/tools/:

1. packages/backend/src/agent/tools/listEmails.ts
 - Tool name: "list_emails"
 - Description: "List emails from the user's inbox with optional filtering by labels or search query"
 - Parameters (use zod for validation):
 - * maxResults: z.number().min(1).max(100).default(10) - Maximum number of emails to return
 - * labelIds: z.array(z.string()).optional() - Filter by label IDs (e.g., ["INBOX", "UNREAD"])
 - * query: z.string().optional() - Gmail search query (e.g., "is:unread from:john@example.com")
 - Returns JSON with count and array of email summaries (id, subject, from, date, snippet)
2. packages/backend/src/agent/tools/searchEmails.ts
 - Tool name: "search_emails"
 - Description: "Search emails using Gmail search syntax. Use this when the user wants to find specific emails."
 - Parameters:
 - * query: z.string() - Gmail search query (required)
 - * maxResults: z.number().min(1).max(50).default(20) - Maximum number of results
 - Returns JSON with query, totalResults, and array of matching emails
3. packages/backend/src/agent/tools/readEmail.ts
 - Tool name: "read_email"
 - Description: "Read the full content of a specific email by its ID, including subject, sender, recipient, and body."
 - Parameters:
 - * emailId: z.string() - The unique ID of the email to read (required)
 - Returns JSON with id, threadId, headers, labels, and body (truncated to 10000 chars)
4. packages/backend/src/agent/tools/sendEmail.ts
 - Tool name: "send_email"
 - Description: "Send an email on behalf of the user. Use this when the user explicitly asks to send an email."
 - Parameters:
 - * to: z.string() - Recipient email address (required)
 - * subject: z.string() - Email subject line (required)
 - * body: z.string() - Email body content in plain text (required)
 - * cc: z.string().optional() - CC recipients (comma-separated)
 - * bcc: z.string().optional() - BCC recipients (comma-separated)
 - * replyToMessageId: z.string().optional() - Message ID to reply to (for threading)
 - Returns JSON with success, messageId, and threadId
5. packages/backend/src/agent/tools/index.ts
 - Create a GmailTool interface with: name, description, schema (ZodObject), execute function
 - Export all tools as an array of GmailTool objects
 - Each execute function should accept (gmail: gmail_v1.Gmail, args: unknown) and return Promise<string>

Then update packages/backend/src/routes/chat.ts to:

- Import the tools from './agent/tools/index.js'

- Register them with `createSdkMcpServer`

Use zod for schema validation. Import `gmail_v1` from '`googleapis`'.

Why Tool Descriptions Matter: Claude decides which tool to use based on descriptions. Be specific about when to use each tool. Notice how `search_emails` says "DO NOT use for general requests" - this prevents Claude from choosing the wrong tool.

Checkpoint

Restart the backend:

```
# Stop the running backend (Ctrl+C) then:  
npm run dev:backend
```

You should still see: Oscar backend running on port 3001

No errors about missing tools or imports.

Troubleshooting

Import errors

- Make sure the `tools/index.ts` exports all tools
- Make sure `chat.ts` imports from the correct path

"tool is not a function"

- Check the import: ``import { tool } from '@anthropic-ai/clause-code'`

Section 5: Set Up Frontend

What we're doing

Creating the React chat interface.

The Prompt

Copy and paste this into Claude Code:

Set up the Oscar frontend. Create these files:

1. packages/frontend/package.json:

```
{  
  "name": "oscar-frontend",  
  "private": true,  
  "type": "module",  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build"  
  },  
  "dependencies": {  
    "react": "^18.3.1",  
    "react-dom": "^18.3.1",  
    "react-markdown": "^9.0.1",  
    "remark-gfm": "^4.0.0",  
    "zustand": "^4.5.5"  
  },  
  "devDependencies": {  
    "@tailwindcss/typography": "^0.5.15",  
    "@types/react": "^18.3.8",  
    "@types/react-dom": "^18.3.0",  
    "@vitejs/plugin-react": "^4.3.1",  
    "autoprefixer": "^10.4.20",  
    "postcss": "^8.4.47",  
    "tailwindcss": "^3.4.12",  
    "typescript": "^5.5.3",  
    "vite": "^5.4.7"  
  }  
}
```
2. packages/frontend/vite.config.ts - Vite config with React plugin and proxy for /api to localhost:3001
3. packages/frontend/tsconfig.json - extends base config, adds jsx: "react-jsx"
4. packages/frontend/tailwind.config.js - includes typography plugin
5. packages/frontend/postcss.config.js - standard PostCSS config
6. packages/frontend/index.html - HTML entry point with title "Oscar - Gmail Assistant"
7. packages/frontend/src/index.css - Tailwind imports
8. packages/frontend/src/main.tsx - React entry point
9. packages/frontend/src/App.tsx - Main app layout with header, ChatContainer, and Sidebar

10. packages/frontend/src/types/index.ts - Message and GmailAccount types
11. packages/frontend/src/store/chatStore.ts - Zustand store for messages, streaming state, session
12. packages/frontend/src/hooks/useAuth.ts - Hook for auth status, login, logout, disconnect
13. packages/frontend/src/hooks/useChat.ts - Hook for sending messages with SSE streaming
14. packages/frontend/src/components/Chat/ChatContainer.tsx - Chat container with message list and input
15. packages/frontend/src/components/Chat/MessageList.tsx - Renders list of messages
16. packages/frontend/src/components/Chat/MessageItem.tsx - Renders single message with markdown support
17. packages/frontend/src/components/Chat/ChatInput.tsx - Input field with send/stop buttons
18. packages/frontend/src/components/Chat/TypingIndicator.tsx - Animated typing dots
19. packages/frontend/src/components/Sidebar/Sidebar.tsx - Shows connected Gmail accounts with add/remove
20. packages/frontend/src/components/Auth/LoginButton.tsx - Login/logout button

Create all files with complete, working code. Use Tailwind CSS for styling.

Checkpoint

Install dependencies and start the frontend:

```
npm install
npm run dev:frontend
```

Open <http://localhost:5173> in your browser.

You should see:

- Oscar header at the top
- "Connect Gmail to get started" message (since no Gmail connected yet)
- Sidebar on the right with Gmail connector

Troubleshooting

Blank page

→ Check browser console for errors (Right-click → Inspect → Console)

"Module not found"

→ Run `npm install` from the project root

Styling looks wrong

→ Make sure Tailwind is configured correctly in tailwind.config.js and postcss.config.js

Section 6: Configure Google OAuth

What we're doing

Setting up Google Cloud Console so Oscar can access Gmail.

The Prompt

Copy and paste this into Claude Code:

```
Help me set up Google OAuth for Oscar. Walk me through these steps one at a time:
```

1. Go to Google Cloud Console (console.cloud.google.com)
2. Create a new project called "Oscar"
3. Enable the Gmail API
4. Configure the OAuth consent screen:
 - User type: External
 - App name: Oscar
 - Add my email as test user
5. Create OAuth credentials:
 - Type: Web application
 - Name: Oscar Web Client
 - Authorized redirect URI: `http://localhost:3001/auth/google/callback`
6. Copy the Client ID and Client Secret

After I complete each step, tell me what to look for to confirm it worked.

Then help me:

7. Generate an encryption key using: `openssl rand -hex 32`
8. Update my .env file with all the values

Don't proceed to the next step until I confirm the current one is done.

Checkpoint

Your .env file should now have real values (not empty):

```
GOOGLE_CLIENT_ID=xxxxxx.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=xxxxxx
GOOGLE_REDIRECT_URI=http://localhost:3001/auth/google/callback
ENCRYPTION_KEY=64-character-hex-string
PORT=3001
FRONTEND_URL=http://localhost:5173
```

Troubleshooting

Can't find Gmail API

→ In Google Cloud Console, go to "APIs & Services" → "Library" → Search "Gmail API"

OAuth consent screen issues

→ Make sure you added yourself as a test user

Encryption key wrong length

→ Run `openssl rand -hex 32` again - it should be exactly 64 characters

Section 7: Test Everything

What we're doing

Running Oscar and connecting Gmail to verify everything works.

The Prompt

Copy and paste this into Claude Code:

```
Help me test Oscar end-to-end:
```

1. First, make sure both servers are stopped (Ctrl+C if running)
2. Start both servers: npm run dev
3. Open http://localhost:5173 in browser
4. Click "Add Gmail account" in the sidebar
5. Complete Google OAuth flow
6. Verify the account appears in the sidebar
7. Try asking: "What emails did I get today?"

Watch the terminal output and tell me if anything looks wrong.

If there are errors, help me debug them one at a time.

Checkpoint

You should be able to:

- [] See Oscar's chat interface
- [] Connect a Gmail account
- [] See your email address in the sidebar
- [] Ask about emails and get real results
- [] See tool indicators when Oscar searches/reads emails

Troubleshooting

"redirect_uri_mismatch"

→ The redirect URI in Google Console must exactly match:
`http://localhost:3001/auth/google/callback`

"No Gmail accounts connected"

→ Try the OAuth flow again. Check that .tokens.json was created.

"Claude Code process exited with code 1"

→ Run `claude login` to re-authenticate

Backend crashes on chat

→ Check the terminal for error messages. Usually a missing import or typo.

Section 8: Add Your Logo (Optional)

What we're doing

Adding a custom logo to Oscar.

The Prompt

Copy and paste this into Claude Code:

```
I want to add a logo to Oscar. Please help me:
```

1. Tell me where to put the logo image (should be packages/frontend/public/)
2. The file should be named panda.png (or tell me how to change the filename in the code)
3. Tell me what size the image should be (recommended dimensions)

```
I'll provide my own image file.
```

Checkpoint

After adding your logo:

1. Refresh the browser
2. You should see your logo in:
 - Browser tab (favicon)
 - Header
 - Chat box header
 - Empty chat state

Troubleshooting

Logo not showing

- Make sure the file is in `packages/frontend/public/`
- Make sure the filename matches what's in the code (default: panda.png)

Section 9: Make It Yours

What we're doing

Customizing Oscar's personality, design, and behavior.

The Prompt

Copy and paste this into Claude Code:

```
Help me customize Oscar. I want to:
```

1. Change the personality/system prompt

Current: "You are Oscar, a helpful AI assistant..."

I want it to be: [DESCRIBE YOUR PREFERRED STYLE - casual, professional, funny, etc.]

2. Show me where to change:

- Colors (Tailwind config)

- The agent's name (if I want to rename it)

- The default empty state message

3. Explain how I would add a new tool if I wanted to (don't create one, just explain)

```
Make the changes I requested and show me how to test them.
```

What to customize

Personality (System Prompt)

Location: packages/backend/src/routes/chat.ts

Examples:

- Casual: "You're a chill email buddy. Keep it short, use emoji sometimes."
- Professional: "You're an executive assistant. Be brief, prioritize important items."
- Helpful: "You're a patient helper. Explain what you're doing, offer suggestions."

Design (Colors/Branding)

Location: packages/frontend/tailwind.config.js

Tool Behavior

Location: packages/backend/src/agent/tools/*.ts

Improve tool descriptions to change when/how Oscar uses them.

The Iteration Cycle

1. **Use it** - Be your own first user
2. **Notice friction** - What's confusing? What's missing?
3. **Adjust** - Change prompts, descriptions, UI
4. **Test** - Verify the changes helped

5. Repeat - This never really ends

Remember: The best agents are refined over time. This is your starting point, not your finish line.

Quick Reference

Starting Oscar

```
npm run dev
```

Stopping Oscar

Press Ctrl+C in the terminal

Key Files

What	Where
Agent personality	`packages/backend/src/routes/chat.ts`
Tools	`packages/backend/src/agent/tools/`
Gmail service	`packages/backend/src/services/gmail.ts`
Frontend components	`packages/frontend/src/components/`
Styling	`packages/frontend/tailwind.config.js`
Environment variables	`.env`

Common Commands

```
# Start everything
npm run dev

# Start only backend
npm run dev:backend

# Start only frontend
npm run dev:frontend

# Kill stuck processes
lsof -ti :3001 | xargs kill -9
lsof -ti :5173 | xargs kill -9

# Re-authenticate Claude
claude login

# Generate new encryption key
openssl rand -hex 32
```

Security Notes

NEVER share or commit:

- Your .env file
- The .tokens.json file
- Your Google Client Secret

The .gitignore is configured to exclude these, but always double-check before sharing code.

Congratulations! You've built Oscar.

Now make it yours.